

DDDDDDDDDDDDDD		CCCCCCCCCCCC	LLL
DDDDDDDDDDDDDD		CCCCCCCCCCCC	LLL
DDDDDDDDDDDDDD		CCCCCCCCCCCC	LLL
DDD	DDD	CCC	LLL
DDD	DDD	CCC	LLL
DDD	DDD	CCC	LLL
DDD	DDD	CCC	LLL
DDD	DDD	CCC	LLL
DDD	DDD	CCC	LLL
DDD	DDD	CCC	LLL
DDD	DDD	CCC	LLL
DDD	DDD	CCC	LLL
DDD	DDD	CCC	LLL
DDD	DDD	CCC	LLL
DDD	DDD	CCC	LLL
DDD	DDD	CCC	LLL
DDDDDDDDDDDDDD		CCCCCCCCCCCC	LLLLLLLLLLLLLLLL
DDDDDDDDDDDDDD		CCCCCCCCCCCC	LLLLLLLLLLLLLLLL
DDDDDDDDDDDDDD		CCCCCCCCCCCC	LLLLLLLLLLLLLLLL

(3)	224	SPAWN COMMAND
(4)	411	SPAWN A SUBPROCESS
(5)	928	PROCESS SPAWN INPUT STREAM
(6)	1098	CHECK FOR ASSOCIATED TERMINAL MAILBOX
(7)	1145	PROCESS SPAWN OUTPUT STREAM
(8)	1283	ATTACH COMMAND
(9)	1340	PERFORM ATTACH OPERATION
(10)	1596	CREATE ATTACH REQUEST MAILBOX
(11)	1649	DELETE ATTACH REQUEST MAILBOX
(12)	1670	CREATE TERMINATION MAILBOX
(13)	1740	DELETE TERMINATION MAILBOX
(14)	1765	DEALLOCATE SPWN BLOCKS
(15)	1794	WRITE RETURNED MESSAGE
(16)	1836	WRITE CONTEXT TO SUBPROCESS
(17)	2088	WRITE RECORD TO CONTEXT MAILBOX
(18)	2115	WRITE ALL SYMBOLS IN A SYMBOL TABLE
(19)	2179	CHECK FOR PENDING HANGUP AST
(20)	2199	GET DEVICE NAME
(21)	2237	CREATE OUTPUT MAILBOX
(22)	2345	DELETE WRITE MAILBOX
(23)	2366	WRITE REQUEST AST FROM A SUBPROCESS
(24)	2436	READ AST FROM READING ATTACH REQUEST RESPONSE
(25)	2468	ATTACH REQUEST AST FROM ANOTHER PROCESS
(26)	2531	SUBPROCESS TERMINATION AST ROUTINE
(27)	2720	BROADCAST NOTIFICATION MESSAGE

```

0000 1
0000 2
0000 3
0000 4
0000 5
0000 6
0000 7
0000 8
0000 9
0000 10
0000 11
0000 12
0000 13
0000 14
0000 15
0000 16
0000 17
0000 18
0000 19
0000 20
0000 21
0000 22
0000 23
0000 24
0000 25
0000 26
0000 27
0000 28
0000 29
0000 30
0000 31
0000 32
0000 33
0000 34
0000 35
0000 36
0000 37
0000 38
0000 39
0000 40
0000 41
0000 42
0000 43
0000 44
0000 45
0000 46
0000 47
0000 48
0000 49
0000 50
0000 51
0000 52
0000 53
0000 54
0000 55
0000 56
0000 57

```

```

      .TITLE SPAWN - MULTI-PROCESSING COMMANDS
      .IDENT 'V04-000'

*****
*
* COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
* DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
* ALL RIGHTS RESERVED.
*
* THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
* ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
* INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
* COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
* OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
* TRANSFERRED.
*
* THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
* AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
* CORPORATION.
*
* DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
* SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
*
*****

AUTHOR:
      Tim Halvorsen, May 1981

MODIFIED BY:
      V03-025 SSA0029      Stan Amway      6-Aug-1984
      Propagate WS default correctly. GETJPI item code was
      JPI$_WSSIZE, should be JPI$_DFWSCNT.

      V03-024 HWS0097      Harold Schultz    01-Aug-1984
      Change max. DCL command string length from 132 to
      WRK C INPBUFSIZ. Change max. logical name translation
      size from 63 to LNMSC_NAMLENGTH.

      V03-023 HWS0088      Harold Schultz    21-Jul-1984
      Fix output mailbox read operation to correctly handle
      an empty mailbox. Fixes problem where subprocesses
      finish but never wake up parent, while running up huge
      buffered I/O counts.

      V03-022 HWS0064      Harold Schultz    02-May-1984
      When attaching to another process, or spawning a new
      new process, don't arm own attach mailbox attention
      AST until just ready to hibernate. (This is to close
      a window where a newly attached process tries to attach
      back to the old current process before it has a chance
      to marked itself detached.)
      Unconceal output stream logical name.

```

0000	58	V03-021	HWS0044	Harold Schultz	30-Mar-1984
0000	59		Add SPAWN/TABLE to allow propagation of table name.		
0000	60		Use table search in cli name verification.		
0000	61		Allow ATTACH to attach to any other process in its		
0000	62		job, removing the the restrictions of having the input		
0000	63		streams be the same for both processes and that the		
0000	64		input device be a terminal.		
0000	65		When spawning a process, indicate to \$CREPRC that a		
0000	66		CLI is being specified. (if none entered via /CLI, the		
0000	67		default cli is specified)		
0000	68		Make WRITE AST use DCL\$SPAWNOUT instead of DCL\$MSGOUT.		
0000	69		Remove \$CHIPDEF		
0000	70				
0000	71	V03-020	HWS0017	Harold Schultz	21-Feb-1984
0000	72		Always signal subprocess errors - do not set inhibit bit.		
0000	73		Use \$BRKTHRUW instead of \$BRDCST.		
0000	74		Change /CLI specification to cli name instead of cli file spec.		
0000	75				
0000	76	V03-019	HWS0002	Harold Schultz	03-Feb-1984
0000	77		Do not propagate the protection mask from a private		
0000	78		logical name table to a sub-process.		
0000	79		Fix logical name hash table processing to process		
0000	80		first bucket in table.		
0000	81				
0000	82	V03-018	PCG0017	Peter Georg	12-Oct-1983
0000	83		Do not set final status if image count has changed.		
0000	84		Check for associated mbx when doing an attach.		
0000	85				
0000	86	V03-017	PCG0016	Peter George	22-Sep-1983
0000	87		Fix bug in null input file processing.		
0000	88				
0000	89	V03-016	PCG0015	Peter George	15-Sep-1983
0000	90		Slap colon at the end of PPD\$T_INPDVI before using.		
0000	91				
0000	92	V03-015	PCG0014	Peter George	16-Aug-1983
0000	93		Correctly supply address of IOSB to IOSM TT_PROCESS QIO.		
0000	94		Be a little more intelligent about when to perform the QIO.		
0000	95				
0000	96	V03-014	PCG0013	Peter George	27-Jun-1983
0000	97		Add /CLI qualifier.		
0000	98		Use event flags more consistently.		
0000	99		Remove code to propagate old format logical name tables.		
0000	100		Specify STSFLG argument to \$CREPRC.		
0000	101		Do an IOSM TT_PROCESS set mode QIO whenever an interactive		
0000	102		process comes back to life.		
0000	103		Remove CTRL/C and out-of-band AST warning message.		
0000	104				
0000	105	V03-013	PCG0012	Peter George	27-Jun-1983
0000	106		Change MOVW in logical name code to MOVL.		
0000	107				
0000	108	V03-012	PCG0011	Peter George	15-Jun-1983
0000	109		Add SPAWN/NOKEYPAD.		
0000	110				
0000	111	V03-011	PCG0010	Peter George	27-May-1983
0000	112		Use DCL\$FORMMSG for logical name message.		
0000	113				
0000	114	V03-010	RAS0157	Ron Schaefer	27-May-1983

0000	115	:	Add support for new logical names to be passed thru
0000	116	:	to the subprocess.
0000	117	:	
0000	118	:	V03-009 PCG0009 Peter George 27-May-1983
0000	119	:	Fix bug in SPAWN CTRL/Y processing.
0000	120	:	Use DCL\$FORMMSG.
0000	121	:	
0000	122	:	V03-008 PCG0008 Peter George 20-Apr-1983
0000	123	:	Add prompt string and keypad state to context that is
0000	124	:	passed through to the subprocess.
0000	125	:	Make spawn work in command procedures.
0000	126	:	
0000	127	:	V03-007 PCG0005 Peter George 30-Mar-1983
0000	128	:	Have SPAWN/NOTIFY use SPWN_T_PROCESS.
0000	129	:	Fix bug in ATTACH_AST that allows illegal attaches.
0000	130	:	Correctly signal process creation errors.
0000	131	:	Process parsed input/output devices as PPF devices.
0000	132	:	
0000	133	:	V03-006 PCG0005 Peter George 29-Mar-1983
0000	134	:	Sort out CTX_C_KEY* symbols.
0000	135	:	
0000	136	:	V03-005 PCG0004 Peter George 01-Mar-1983
0000	137	:	Call DCL\$GETNVAL. Propagate keypad symbols to subprocess.
0000	138	:	
0000	139	:	V03-004 PCG0003 Peter George 01-Feb-1983
0000	140	:	Clean up code, fix bugs.
0000	141	:	Add /SYMBOLS, /LOGICAL_NAMES, /NOTIFY.
0000	142	:	
0000	143	:	V03-003 PCG0002 Peter George 14-Jan-1983
0000	144	:	Fix SPWN deallocation problem.
0000	145	:	
0000	146	:	V03-002 PCG0001 Peter George 18-Nov-1982
0000	147	:	Store size of allocated TMBX structure in that structure.
0000	148	:	
0000	149	:	V03-001 KDM0002 Kathleen D. Morse 28-Jun-1982
0000	150	:	Added \$IODEF, \$PSLDEF and \$SSDEF.
0000	151	:	
0000	152	:---	

```

0000 154 : MACRO LIBRARY CALLS
0000 155 :
0000 156 :
0000 157 :SPRCDEF ;DEFINE CREPRC FLAGS
0000 158 :PRCDEF ;DEFINE PROCESS WORK AREA
0000 159 :WRKDEF ;DEFINE COMMAND WORK AREA
0000 160 :PTRDEF ;DEFINE TOKEN DESCRIPTORS
0000 161 :SYMDEF ;DEFINE SYMBOL TABLE ENTRY
0000 162 :SPWNDEF ;DEFINE SPAWN LOCAL STORAGE
0000 163 :CTXDEF ;DEFINE PROCESS CONTEXT MESSAGES
0000 164 :TMBXDEF ;DEFINE TERMINATION MAILBOX STRUCTURE
0000 165 :SBRKDEF ;DEFINE BREAKTHRU CLASSES
0000 166 :SPPDDEF ;DEFINE PPD FIELDS
0000 167 :SDIBDEF ;DEFINE GETDEV INFO BUFFER
0000 168 :$JPIDEF ;DEFINE GETJPI ITEM CODES
0000 169 :$DVIDEF ;DEFINE GETDVI ITEM CODES
0000 170 :$SYIDEF ;DEFINE GETSYI ITEM CODES
0000 171 :$PQLDEF ;DEFINE PROCESS QUOTA TYPE CODES
0000 172 :$PSLDEF ;DEFINE PROGRAM STATUS LONGWORD FIELDS
0000 173 :$IODEF ;DEFINE I/O FUNCTION CODES
0000 174 :$SSDEF ;DEFINE SYSTEM STATUS CODES
0000 175 :$LOGDEF ;DEFINE LOGICAL NAME ENTRY
0000 176 :$LNMDEF ;DEFINE LOGICAL NAME ATTRIBUTES
0000 177 :$LNMSTRDEF ;DEFINE LOGICAL NAME STRUCTURES
0000 178 :$ACCDDEF ;TERMINATION RECORD FORMAT
0000 179 :$CLMSGDEF ;CLI MESSAGE CODES
0000 180 :$DEVDEF ;DEFINE DEVICE CHARACTERISTICS
0000 181 :$FABDEF ;DEFINE FAB FIELDS
0000 182 :$NAMDEF ;DEFINE NAM FIELDS
0000 183 :$TT2DEF ;DEFINE DEVDEPEND2 FIELDS
0000 184 :$DCDEF ;DEFINE DVI DEVICE CLASSES
0000 185 :$$CLITABDEF ;DEFINE MAX PROMPT SIZE
0000 186 :
00000000 187 .PSECT DCL$ZCODE,BYTE,RD,NOWRT
0000 188 :
00000010 0000 189 ATTMBX_MAXMSG = 16 ; MAXIMUM SIZE OF ATTACH REQUEST MESSAGE
0000 190 :
0000 191 LOGINOUT:
0000 192 .ASCIC 'SYSS$SYSTEM:LOGINOUT' ; IMAGE TO INITIALIZE DCL
3A 4D 45 54 53 59 53 24 53 59 53 00' 0000
54 55 4F 4E 49 47 4F 4C 000C
13 0000
0014 193 :
0014 194 PRCNAM_NAME:
0014 195 .ASCIC '!AS_!UL' ; FAO STRING FOR DEFAULT PROCESS NAME
0014 196 :
001C 196 ATTACH_NAME:
001C 197 .ASCIC 'DCL$ATTACH_!XL' ; FAO STRING FOR ATTACH LOGNAME
5F 48 43 41 54 54 41 24 4C 43 44 00' 001C
4C 58 21 002B
0E 001C
002B 198 OUTPUT_NAME:
002B 199 .ASCIC 'DCL$OUTPUT_!XL' ; FAO STRING FOR OUTPUT LOGNAME
5F 54 55 50 54 55 4F 24 4C 43 44 00' 002B
4C 58 21 0037
0E 002B
003A 200 :
003A 201 SYSS$INPUT:
003A 202 .ASCIC 'SYSS$INPUT' ; DEFAULT INPUT STREAM
54 55 50 4E 49 24 53 59 53 00' 003A
09 003A

```

54 55 50 54 55 4F 24 53 59 53 00'	0044	203	SYSS\$OUTPUT:		
	0A 0044	204	.ASCIC	'SYSS\$OUTPUT'	; DEFAULT OUTPUT STREAM
3A 4D 45 54 53 59 53 24 53 59 53 00'	004F	205	SYSS\$SYSTEM:		
	0B 004F	206	.ASCIC	'SYSS\$SYSTEM:'	; DEVICE SPEC. FOR CLI SPECIFICATION
	005B	207	NL:		
3A 4C 4E 5F 00'	005B	208	.ASCIC	'_NL:'	; NULL DEVICE
	04 005B				
	0060	209	COM:		
4D 4F 43 2E 00'	0060	210	.ASCIC	'_COM'	; DEFAULT INPUT FILE TYPE
	04 0060				
	0065	211	LOG:		
47 4F 4C 2E 00'	0065	212	.ASCIC	'_LOG'	; DEFAULT OUTPUT FILE TYPE
	04 0065				
	006A	213			
	006A	214	CLI_NAME:		; KNOWN CLI NAME TABLE.
4C 43 44 00'	006A	215	.ASCIC	'DCL'	; CLI NAME FOR DCL
	03 006A				
52 43 4D 00'	006E	216	.ASCIC	'MCR'	; CLI NAME FOR MCR
	03 006E				
4C 4C 45 48 53 00'	0072	217	.ASCIC	'SHELL'	; CLI NAME FOR SHELL
	05 0072				
	00 0078	218	.BYTE	0	; END OF TABLE MARKER
	0079	219			
73 65 63 6F 72 70 62 75 53 07 07 00'	0079	220	NOTIFY_MSG:		
6F 63 20 73 61 68 20 43 41 21 20 73	0085	221	.ASCIC	'Subprocess !AC has completed'	
	0091				
	1E 0079				
0000002C	0098	222		NOTIFY_LEN = 28+15+1	

```

0098 224 .SBTTL SPAWN COMMAND
0098 225
0098 226 :+ DCL$SPAWN - SPAWN COMMAND
0098 227
0098 228 THIS ROUTINE IS CALLED TO EXECUTE THE DCL SPAWN COMMAND. THE SPAWN
0098 229 COMMAND CREATES A "CLONED" SUBPROCESS WITH THE FOLLOWING CONTEXT COPIED
0098 230 FROM THE PARENT TO THE SUBPROCESS:
0098 231
0098 232 1) ALL CLI SYMBOLS
0098 233 2) ALL PROCESS LOGICAL NAMES
0098 234 3) DEFAULT DISK AND DIRECTORY
0098 235 4) CURRENT PROCESS PRIVILEGES
0098 236 5) CURRENT COMMAND VERIFICATION STATE
0098 237 6) CURRENT "ON CONTROL" (OUT-OF-BAND) STATE
0098 238 7) CURRENT PROMPT STRING
0098 239 8) CURRENT KEYPAD STATE
0098 240
0098 241 NO PROCESS PERMANENT OPEN FILES ARE COPIED, NOR IS ANY IMAGE OR PROCEDURE
0098 242 CONTEXT. THE SUBPROCESS IS SET TO PROCEDURE LEVEL 0. LOGIN.COM IS NOT
0098 243 EXECUTED, BOTH BECAUSE THE CONTEXT IS COPIED SEPARATELY AND TO CAUSE THE
0098 244 SUBPROCESS TO INITIALIZE QUICKLY. THE PARENT IS LEFT IN HIBERNATION STATE
0098 245 UNTIL THE SUBPROCESS TERMINATES OR TRANSFERS CONTROL BACK TO THE PARENT VIA
0098 246 THE ATTACH COMMAND.
0098 247
0098 248 INPUTS:
0098 249
0098 250 R10 = ADDRESS OF COMMAND WORK AREA
0098 251 R11 = ADDRESS OF PROCESS WORK AREA
0098 252
0098 253
0098 254 OUTPUTS:
0098 255
0098 256 R0 = STATUS CODE
0098 257
0098 258 R2-R9 DESTROYED.
0098 259
0098 260
0098 261 DCL$SPAWN::
0098 262
0098 263 : ALLOCATE SOME SPACE FOR SPAWN STORAGE
0098 264
0098 265 MOVZWL #SPWN_C_LENGTH,R1 ; LENGTH OF STORAGE TO ALLOCATE
0098 266 BSBW DCL$ACLDYNMEM ; ALLOCATE STORAGE
0098 267 BLBS R0,10$ ; BRANCH IF OK
0098 268 RSB ; IF ERROR DETECTED, REPORT IT
0098 269 10$: MOVL R2,R6 ; POINT TO SPWN STORAGE
0098 270 MOVCS #0,(R1),#0,#SPWN_C_LENGTH,(R6) ; ZERO THE BLOCK
0098 271 ; (WITHOUT DESTROYING R1)
0098 272 MOVW R1,SPWN_W_SIZE(R6) ; STORE SIZE OF BLOCK
0098 273 BISW #SPWN_M_LOG!SPWN_M_WAIT!- ; ASSUME /LOG, /WAIT
0098 274 SPWN_M_CLISYM!SPWN_M_LOGNAM!- ; COPY CLI SYMBOLS & LOGNAMES
0098 275 SPWN_M_KEYPAD,SPWN_W_FLAGS(R6)
0098 276 BICW #SPWN_M_CLI!SPWN_M_TABLE,- ; ASSUME /NOCLI AND /NOTABLE
0098 277 SPWN_W_FLAGS(R6)
0098 278 MOVW DCL$CRCF,SPWN_W_PMPTCTRL(R6) ; ASSUME /CONTROL
0098 279 MNEGB #1,SPWN_W_EFNTR6) ; DO NOT SET EVENT FLAG ON TERMINATION
0098 280

```

```

00CC 281 :
00CC 282 : PROCESS THE VERB QUALIFIERS ON THE COMMAND LINE
00CC 283 :
FF31' 30 00CC 284 40$: BSBW DCL$GETDVAL : GET NEXT TOKEN
10 50 E9 00CF 285 R0,43$ : IF EOL, END OF PROCESSING
00 55 D1 00D2 286 CMPL R5,#PTR_K_COMDQUAL : VERB QUALIFIER?
0E 13 00D3 287 BEQL 42$ : BRANCH IF SO
03 55 D1 00D7 288 CMPL R5,#PTR_K_PARAMETR : PARAMETER (COMMAND STRING)?
FO 12 00DA 289 BNEQ 40$ : IF NOT, IGNORE IT
30 A6 51 7D 00DC 290 MOVQ R1,SPWN_Q_CMDSTR(R6) : SAVE DESCRIPTOR OF COMMAND STRING
EA 11 00E0 291 BRB 40$
0171 31 00E2 292 43$: BRW 80$ : END OF COMMAND PARSING
FF18' 30 00E3 293 42$: BSBW DCL$GETNVAL : GET QUALIFIER NUMBER
00'8F 51 91 00E8 294 CMPB R1,#CLISK_SPAW_WAIT : /WAIT?
27 13 00EC 295 BEQL 49$
00'8F 51 91 00EE 296 CMPB R1,#CLISK_SPAW_LOG : /LOG?
2F 13 00F2 297 BEQL 47$
00'8F 51 91 00F4 298 CMPB R1,#CLISK_SPAW_SYMB : /SYMBOLS?
37 13 00F8 299 BEQL 50$
00'8F 51 91 00FA 300 CMPB R1,#CLISK_SPAW_LOGI : /LOGICAL_NAMES?
3F 13 00FE 301 BEQL 51$
00'8F 51 91 0100 302 CMPB R1,#CLISK_SPAW_NOTI : /NOTIFY?
49 13 0104 303 BEQL 52$
00'8F 51 91 0106 304 CMPB R1,#CLISK_SPAW_CARR : /CARRIAGE_CONTROL?
53 13 010A 305 BEQL 531$
00'8F 51 91 010C 306 CMPB R1,#CLISK_SPAW_KEYP : /KEYPAD?
60 13 0110 307 BEQL 532$
006D 31 0112 308 BRW 60$ : CHECK OTHER QUALIFIER POSSIBILITIES
0115 309
0115 310 49$: SETBIT SPWN_V_WAIT,SPWN_W_FLAGS(R6) : ASSUME /WAIT
AF 53 00 E1 0119 311 BBC #PTR_V_NEGATE-PTR_V_FLAGS,R3,40$ : IGNORE IF NOT /NOWAIT
A9 11 011D 312 CLRBIT SPWN_V_WAIT,SPWN_Q_FLAGS(R6) : CLEAR WAIT FLAG
0121 313 BRB 40$
0123 314 47$: SETBIT SPWN_V_LOG,SPWN_W_FLAGS(R6) : ASSUME /LOG
A1 53 00 E1 0127 315 BBC #PTR_V_NEGATE-PTR_V_FLAGS,R3,40$ : IGNORE IF NOT /NOLOG
012B 316 CLRBIT SPWN_V_LOG,SPWN_W_FLAGS(R6) : CLEAR LOG MESSAGE FLAG
9B 11 012F 317 BRB 40$
0131 318 50$: SETBIT SPWN_V_CLISYM,SPWN_W_FLAGS(R6) : ASSUME /SYMBOLS
93 53 00 E1 0135 319 BBC #PTR_V_NEGATE-PTR_V_FLAGS,R3,40$ : IGNORE IF NOT /NOSYMBOLS
0139 320 CLRBIT SPWN_V_CLISYM,SPWN_Q_FLAGS(R6) : CLEAR CLISYM FLAG
8D 11 013D 321 BRB 40$
013F 322 51$: SETBIT SPWN_V_LOGNAM,SPWN_W_FLAGS(R6) : ASSUME /LOGICAL_NAMES
5E 53 00 E1 0144 323 BBC #PTR_V_NEGATE-PTR_V_FLAGS,R3,41$ : IGNORE IF NOT /NOLOGICAL_NAMES
0148 324 CLRBIT SPWN_V_LOGNAM,SPWN_Q_FLAGS(R6) : CLEAR LOGNAM FLAG
57 11 014D 325 BRB 41$
014F 326 52$: SETBIT SPWN_V_NOTIFY,SPWN_W_FLAGS(R6) : ASSUME /NOTIFY
4E 53 00 E1 0154 327 BBC #PTR_V_NEGATE-PTR_V_FLAGS,R3,41$ : IGNORE IF NOT /NONOTIFY
0158 328 CLRBIT SPWN_V_NOTIFY,SPWN_Q_FLAGS(R6) : CLEAR NOTIFY FLAG
47 11 015D 329 BRB 41$
00A3 C6 00000000'EF B0 015F 330 531$: MOVW DCL$CRLF,SPWN_W_PMPTCTRL(R6) : ASSUME /CONTROL
3A 53 00 E1 0168 331 BBC #PTR_V_NEGATE-PTR_V_FLAGS,R3,41$ : IGNORE IF NOT /NOCONTROL
00A3 C6 B4 016C 332 CLRW SPWN_W_PMPTCTRL(R6) : SET /NOCONTROL
34 11 0170 333 BRB 41$
0172 334 532$: SETBIT SPWN_V_KEYPAD,SPWN_W_FLAGS(R6) : ASSUME /KEYPAD
2B 53 00 E1 0177 335 BBC #PTR_V_NEGATE-PTR_V_FLAGS,R3,41$ : IGNORE IF NOT /NOKEYPAD
017B 336 CLRBIT SPWN_V_KEYPAD,SPWN_Q_FLAGS(R6) : CLEAR KEYPAD FLAG
24 11 0180 337 BRB 41$

```

00'8F	51	91	0182	338					
	51	13	0182	339	608:	CMPB	R1,#CLISK_SPAW_INPU	:	/INPUT?
00'8F	51	13	0186	340		BEQL	458		
	51	91	0188	341		CMPB	R1,#CLISK_SPAW_OUTP	:	/OUTPUT?
	51	13	018C	342		BEQL	468		
00'8F	51	91	018E	343		CMPB	R1,#CLISK_SPAW_PROG	:	/PROCESS_NAME?
	51	13	0192	344		BEQL	488		
00'8F	51	91	0194	345		CMPB	R1,#CLISK_SPAW_CLI	:	/CLI?
	51	13	0198	346		BEQL	4818		
00'8F	51	91	019A	347		CMPB	R1,#CLISK_SPAW_TABL	:	/TABLE?
	63	13	019E	348		BEQL	4858		
00'8F	51	91	01A0	349		CMPB	R1,#CLISK_SPAW_PROM	:	/PROMPT?
	75	13	01A4	350		BEQL	538		
	FF23	31	01A6	351	418:	BRW	408	:	IF NONE OF THE ABOVE QUALS, IGNORE IT
			01A9	352					
			01A9	353	458:	CLRBIT	SPWN_V_INPUT,SPWN_W_FLAGS(R6)	:	ASSUME NO VALUE PRESENT
02	54	91	01AE	354		CMPB	R4,#PTR_K_COLON	:	IS A VALUE PRESENT?
	F3	12	01B1	355		BNEQ	418	:	IF NOT, USE DEFAULT
			01B3	356		SETBIT	SPWN_V_INPUT,SPWN_W_FLAGS(R6)	:	INDICATE VALUE PRESENT
	FE45'	30	01B8	357		BSBW	DCL\$GETDVAL	:	GET /INPUT VALUE
20	A6	51	01BB	358		MOVQ	R1,SPWN_Q_INPUT(R6)	:	USE EXPLICIT INPUT STREAM
	E5	11	01BF	359		BRB	418		
			01C1	360	468:	CLRBIT	SPWN_V_OUTPUT,SPWN_W_FLAGS(R6)	:	ASSUME NO VALUE PRESENT
02	54	91	01C6	361		CMPB	R4,#PTR_K_COLON	:	IS A VALUE PRESENT?
	DB	12	01C9	362		BNEQ	418	:	IF NOT, USE DEFAULT
			01CB	363		SETBIT	SPWN_V_OUTPUT,SPWN_W_FLAGS(R6)	:	INDICATE VALUE PRESENT
	FE2D'	30	01D0	364		BSBW	DCL\$GETDVAL	:	GET /OUTPUT VALUE
28	A6	51	01D3	365		MOVQ	R1,SPWN_Q_OUTPUT(R6)	:	USE EXPLICIT OUTPUT STREAM
	CD	11	01D7	366		BRB	418		
	02	54	01D9	367	488:	CMPB	R4,#PTR_K_COLON	:	IS A VALUE PRESENT?
	C8	12	01DC	368		BNEQ	418	:	IF NOT, USE DEFAULT
	FE1F'	30	01DE	369		BSBW	DCL\$GETDVAL	:	GET VALUE
18	A6	51	01E1	370		MOVQ	R1,SPWN_Q_PROCNAM(R6)	:	USE EXPLICIT PROCESS NAME
			01E5	371		SETBIT	SPWN_V_PROCNAM,SPWN_W_FLAGS(R6)	:	SET PROCESS NAME SPECIFIED
	BB	12	01E9	372		BNEQ	418		
			01EB	373	4818:	CLRBIT	SPWN_V_CLI,SPWN_W_FLAGS(R6)	:	ASSUME /NOCLI
B2	53	00	01F0	374		BBS	#PTR_V_NEGATE-PTR_V_FLAGS,R3,418	:	FLAG OK AS IS IF /NOCLI
			01F4	375		SETBIT	SPWN_V_CLI,SPWN_W_FLAGS(R6)	:	SET /CLI PRESENT FLAG
	FE04'	30	01F9	376		BSBW	DCL\$GETDVAL	:	GET /CLI VALUE
00C6	C6	51	01FC	377		MOVQ	R1,SPWN_Q_CLI(R6)	:	SAVE DESCRIPTOR OF CLI NAME
	A3	11	0201	378		BRB	418		
			0203	379	4858:	CLRBIT	SPWN_V_TABLE,SPWN_W_FLAGS(R6)	:	ASSUME /NOTABLE
9A	53	00	0208	380		BBS	#PTR_V_NEGATE-PTR_V_FLAGS,R3,418	:	FLAG OK AS IS IF /NOTABLE
			020C	381		SETBIT	SPWN_V_TABLE,SPWN_W_FLAGS(R6)	:	SET /TABLE PRESENT FLAG
	FDEC'	30	0211	382		BSBW	DCL\$GETDVAL	:	GET /TABLE VALUE
00CE	C6	51	0214	383		MOVQ	R1,SPWN_Q_TABLE(R6)	:	SAVE DESCRIPTOR OF TABLE NAME
	88	11	0219	384		BRB	418		
			021B	385	538:	CLRBIT	SPWN_V_PROMPT,SPWN_W_FLAGS(R6)	:	ASSUME /NOPROMPT
	30	53	0220	386		BLBS	R3,538	:	BRANCH IF SO
			0223	387		SETBIT	SPWN_V_PROMPT,SPWN_W_FLAGS(R6)	:	SET /PROMPT
	00'8F	90	0228	388		MOVQ	#DCL\$C_PROMPTLEN,-	:	ASSUME NO VALUE SPECIFIED
	00A2	C6	022B	389			SPWN_B_PROMPTLEN(R6)	:	
	54	DD	022E	390		PUSHL	R4	:	SAVE THE TERMINATOR
00000000'EF	0000'8F	28	0230	391		MOVQ	#DCL\$C_PROMPTLEN,DCL\$CRLF,-	:	GET PROMPT STRING
	00A3	C6	0239	392			SPWN_W_PROMPTCTRL(R6)	:	
	54	8ED0	023C	393		POPL	R4	:	RESTORE THE TERMINATOR
	02	54	023F	394		CMPB	R4,#PTR_K_COLON	:	IS A VALUE PRESENT?

```

00A2 C6      0F 12 0242 395      BNEQ 5338      : USE DEFAULT PROMPT
              FDB9 30 0244 396      BSBW DCL$GETDVAL : GET VALUE
              51 03 81 0247 397      ADDB3 #3,R1,SPWN_B_PROMPTLEN(R6) : GET PROMPT LENGTH
              62 20 28 024D 398      MOVC3 #ENT_K_MAX_PROMPT,(R2),- : GET PROMPT STRING
              00A6 C6 0250 399      SPWN_G_PROMPT(R6)
              FE76 31 0253 400 5338: BRW 40$
              0256 401
              0256 402
              0256 403 : THE PARSING IS COMPLETE, AND ALL OPTIONS ARE SET INTO THE SPWN
              0256 404 : BLOCK. NOW PERFORM THE SPAWN OPERATION.
              0256 405
              07 10 0256 406 80$: BSBW DCL$SPAWN2 : PERFORM THE SPAWN OPERATION
              03 50 E9 0258 407      BLBC R0, 90$ : TEST FOR FAILURE OF SPAWN ITSELF
              50 51 D0 025B 408      MOVL R1, R0 : GET SUBPROCESS TERMINATION STATUS
              05 025E 409 90$: RSB

```

```

025F 411 .SBTTL SPAWN A SUBPROCESS
025F 412 ---
025F 413
025F 414 THIS ROUTINE IS CALLED TO PERFORM THE ACTUAL CREATION OF THE SUBPROCESS.
025F 415
025F 416 INPUTS:
025F 417
025F 418 R6 = ADDRESS OF SPWN PARAMETER BLOCK
025F 419 R11 = ADDRESS OF PRC AREA
025F 420
025F 421 OUTPUTS:
025F 422
025F 423 R0 = STATUS
025F 424 R1 = FINAL STATUS FROM COMPLETED SUBPROCESS
025F 425 SPWN_L_PID(R6) = PID OF SUBPROCESS (IF NOWAIT FLAG SPECIFIED)
025F 426 ---
025F 427
01BC 8F BB 025F 428 DCL$SPAWN2::
025F 429 PUSHF #M<R2,R3,R4,R5,R7,R8> ; SAVE REGISTERS
0263 430
0263 431
0263 432 CONTROL/Y AST'S ARE DISABLED THROUGHOUT THIS COMMAND, TO ENSURE THAT
0263 433 MAILBOXES WHICH ARE CREATED HERE ARE CORRECTLY DELETED, ETC. AS A
0263 434 RESULT, WE MUST PERIODICALLY CHECK THE HANGUP FLAG IN CASE A HANGUP
0263 435 IS DETECTED WHILE WE ARE OPERATING.
0263 436
00B4 CB DO 0263 437 MOVL PRC_L_OUTOFBAND(R11),- ; SAVE OUT-OF-BAND ENABLE MASK
58 A6 0267 438 SPWN_L_OUTOFBAND(R6)
51 D4 0269 439 CLRL R1 ; DISABLE ALL OUT-OF-BAND AST'S
FD92' 30 026B 440 BSBW DCL$RESET00B
026E 441
026E 442
026E 443 PRE-CLEAR THE TERMINATION EVENT FLAG, IF ONE IS SPECIFIED.
026E 444
0F A6 95 026E 445 TSTB SPWN_B_EFN(R6) ; EVENT FLAG SPECIFIED?
0A 19 0271 446 BLSS 2$ ; BRANCH IF NONE
0273 447 SCLREF_S EFN=SPWN_B_EFN(R6) ; PRE-CLEAR IT
027D 448
027D 449
027D 450 SAVE THE PREVIOUS ACCESS MODE, FOR PROBING RETURN STATUS AND DELIVERING
027D 451 TERMINATION ASTS.
027D 452
027D 453 2$: MOVPSL R0 ; GET CURRENT PSL
027F 454 EXTZV #PSL$V_PVMOD,#PSL$S_PVMOD,- ; EXTRACT PREVIOUS ACMODE
0282 455 R0,R0
0284 456 MOVB R0,SPWN_B_ACMODE(R6) ; SAVE THE ACCESS MODE
0288 457
0288 458
0288 459 ALLOCATE SOME SCRATCH STORAGE ON THE STACK (1 - 1/2 PAGES)
0288 460
0288 461 MOVAB -768(SP),SP ; ALLOCATE SCRATCH BUFFER ON STACK
028D 462 PUSHL SP ; CONSTRUCT DESCRIPTOR OF BUFFER
028F 463 MOVZWL #768,-(SP)
0294 464 MOVL SP,R8 ; AND POINT TO IT
0297 465
0297 466
0297 467 FIND A TERMINATION MAILBOX SLOT AND INITIALIZE IT FOR THIS SUBPROCESS.

```

```

0297 468 :
0297 469 :
57 74 AB 9E 0297 470 : ASSUME TMBX_L LINK EQ 0
57 67 D0 0297 471 10$: MOVAB PRC_C_TMBX(R11),R7
09 12 0298 472 : GET ADDRESS OF TMBX BLOCK LIST HEA
OACE 30 029E 473 : GET NEXT BLOCK IN LIST
03 50 E8 02A0 474 : BRANCH IF NOT END OF LIST
00FA 31 02A3 475 : CREATE NEW MBX IF END OF LIST
04 08 A7 91 02A6 476 20$: BRW RO,20$
EC 13 02A9 477 : CONTINUE IF NO ERROR
08 A7 96 02AD 478 : EXIT IF ERROR
08 A6 04 A7 B0 02AF 479 : DOES MBX HAVE ANY OPEN SLOTS?
02B2 480 : NO, LOOK AT NEXT MBX IN LIST
02B7 481 : YES, GRAB ONE, INCR REFERENCE COUN
02B7 482 : COPY THE UNIT NUMBER
02B7 483 :
02B7 484 : CHECK FOR KNOWN CLI'S TO WHICH TO PASS CONTEXT IF /CLI
02BA 485 : WAS SPECIFIED.
02BF 486 :
02C4 487 :
02C9 488 :
59 68 9A 02C9 489 21$: MOVZBL (R8),R9
16 13 02CC 490 : EXTRACT LENGTH
02CE 491 : IF EQ, SPECIFIED CLI NOT KNOWN
02CE 492 :
00 7E 52 7D 02CE 493 : SAVE DESCRIPTOR
63 52 2D 02D1 494 : COMPARE TO THIS TABLE ENTRY
01 AB 59 13 02D5 495 :
11 13 02D8 496 : IF EQ, THIS CLI IS KNOWN
02DA 497 :
52 8E 7D 02DA 498 : RESTORE DESCRIPTOR
01 A849 9E 02DD 499 : SET UP TO CHECK NEXT ENTRY
E5 11 02E2 500 : CHECK NEXT TABLE ENTRY
02E4 501 :
02E4 502 40$: SETBIT SPWN_V_NOCTX,SPWN_W_FLAGS(R6) : DON'T SEND CONTEXT TO FOR THIS CLI
52 03 11 02E9 503 :
8E 7D 02EB 504 45$: BRB 50$ : RESTORE STACK
02EE 505 :
02EE 506 : PROPAGATE EITHER THE SPECIFIED CLI AND TABLE NAMES OR THE DEFAULT NAMES TO
02EE 507 : THE SPAWN CLI AND TABLE SPECIFICATION TABLES.
58 00C6 C6 3C 02EE 508 50$: MOVZWL SPWN_Q_CLI(R6),R8 : GET LENGTH OF CLI NAME, IF SPEC.
59 00CE C6 3C 02F3 509 : MOVZWL SPWN_Q_TABLE(R6),R9 : GET LENGTH OF TABLE NAME, IF SPECI
40 0C A6 0F E1 02F8 510 : BBC #SPWN_Q_TABLE,SPWN_W_FLAGS(R6),218$ : SKIP IF NO TABLE SPECIFIED
24 0C A6 0D E1 02FD 511 : BBC #SPWN_V_CLI,SPWN_W_FLAGS(R6),216$ : SKIP IF NO CLI SPECIFIED
0302 512 :
0302 513 : BOTH CLI AND NAME WERE SPECIFIED
0302 514 :
0302 515 :
00000000'GF 58 90 0302 516 : MOVB R8,G^CTL$GT SPAWNCLI : SET SIZE FIELD
00CA D6 58 28 0309 517 : MOVG R8,SPWN_Q_CLI+4(R6),- : MOVE CLI SPECIFICATION
00000001'GF 59 90 030E 518 : MOVG R9,G^CTL$GT SPAWNTABLE : SET LENGTH OF TABLE NAME
00000000'GF 59 28 0313 519 : MOVG R9,SPWN_Q_TABLE+4(R6),- : COPY TABLE NAME
00D2 D6 59 28 031A 520 : MOVG R9,G^CTL$GT-SPAWNTABLE+1
00000001'GF 5F 11 031F 521 : BRB 228$
0324 522 :
0326 523 : JUST TABLE NAME SPECIFIED
0326 524 :

```

```

58 00000000'GF 9A 0326 525 216$: MOVZBL G^CTL$GT_CLNAME,R8 ; GET LENGTH OF CLI NAME
                                58 D6 032D 526 ; INCL R8 ; INCLUDE LENGTH BYTE
00000000'GF 58 28 032F 527 ; MOV R8,G^CTL$GT_CLNAME,- ; GET DEFAULT CLI NAME
00000000'GF 58 0336 528 ; G^CTL$GT_SPAWNCLI
                                D6 11 033B 529 ; BRB 214$ ; NOW TRANSER TABLE NAME
                                033D 530 ;
                                033D 531 ; NO TABLE NAME SPECIFIED. CHECK IF CLI NAME SPECIFIED
                                033D 532 ;
                                033D 533 218$: BBC #SPWN V CLI,SPWN W FLAGS(R6),220$ ; SKIP IF CLI NOT SPECIFIED
19 0C A6 0D E1 033D 534 ; CLR R8 ; G^CTL$GT_SPAWNTABLE ; ZERO TABLE SPECIFICATION
00000000'GF 58 94 0342 535 ; MOV R8,G^CTL$GT_SPAWNCLI ; SET SIZE FIELD
00000000'GF 58 90 0348 536 ; MOV R8,SPWN Q CLI+4(R6),- ; MOVE CLI SPECIFICATION
00CA D6 58 28 034F 537 ; G^CTL$GT_SPAWNCLI+1
00000001'GF 58 0354 538 ; BRB 228$
2A 11 0359 539 ;
                                035B 540 ; NEITHER CLI OR TABLE SPECIFIED. JUST COPY DEFAULTS
                                035B 541 ;
05 00000000'GF 9A 035B 542 220$: MOVZBL G^CTL$GT_CLNAME,R8 ; GET LENGTH OF CLI NAME
                                58 D6 0362 543 ; INCL R8 ; INCLUDE LENGTH BYTE
00000000'GF 58 28 0364 544 ; MOV R8,G^CTL$GT_CLNAME,- ; COPY CLI NAME
00000000'GF 58 036B 545 ; G^CTL$GT_SPAWNCLI
58 00000000'GF 9A 0370 546 ; MOVZBL G^CTL$GT_TABLENAME,R8 ; GET LENGTH OF TABLE NAME
                                58 D6 0377 547 ; INCL R8 ; INCLUDE LENGTH BYTE
00000000'GF 58 28 0379 548 ; MOV R8,G^CTL$GT_TABLENAME,- ; COPY TABLE NAME
00000000'GF 58 0380 549 ; G^CTL$GT_SPAWNTABLE
                                58 8E 7D 0385 550 228$: MOVQ (SP)+,R8 ; RESTORE WORK REG.
                                0388 551 ;
                                0388 552 ; CREATE A COMMUNICATIONS MAILBOX TO COPY THE CONTEXT TO THE SUBPROCESS
                                0388 553 ;
5F 0C A6 0E E0 0388 554 ; BBS #SPWN V NOCTX,SPWN W FLAGS(R6),22$ ; BRANCH IF UNKNOWN CLI WAS SPECIF
                                038D 555 ; $CREMBX,S CHAN=SPWN W_CHAN(R6),- ; CREATE COMMUNICATIONS MAILBOX
                                038D 556 ; MAXMSG=#CTX_C_MAXLEN
                                4C 50 E9 03A3 557 230$: BLBC R0,95$ ; BRANCH ON ERROR
                                03A6 558 ;
                                03A6 559 ; GET THE NAME OF THAT MAILBOX USING $GETDVI.
                                03A6 560 ;
                                03A6 561 ;
                                03A6 562 ;
                                10 7E D4 03A6 563 ; CLRL -(SP) ; CREATE GETDVI ITEM LIST
                                7E 10 9F 03A8 564 ; PUSHAB SPWN Q MBXNAM(R6) ; ADDRESS OF WORD TO RECEIVE LENGTH
                                7E 68 7D 03AB 565 ; MOVQ (R8)-,TSP ; SET DESCRIPTOR OF BUFFER
02 AE 20 B0 03AE 566 ; MOVW #DVI$DEVNAM,2(SP) ; REQUEST COMPLETE DEVICE NAME
50 5E D0 03B2 567 ; MOVL SP,R0 ;
                                03B5 568 ; $GETDVIW,S CHAN=SPWN W_CHAN(R6),- ; GET MAILBOX DEVICE NAME
                                03B5 569 ; IOSB=SPWN Q IOSB(R6),-
                                03B5 570 ; EFN=#EXEFC SYSEFN,-
                                03B5 571 ; ITMLST=(R0)
                                SE 10 C0 03D1 572 ; ADDL #4*4,SP ; POP GETDVI ITEM LIST
                                1B 50 E9 03D4 573 ; BLBC R0,95$ ; BRANCH IF ERROR DETECTED
50 38 A6 3C 03D7 574 ; MOVZWL SPWN Q IOSB(R6),R0 ; GET IOSB STATUS
14 50 E9 03DB 575 ; BLBC R0,95$ ; BRANCH IF ERROR DETECTED
                                03DE 576 ;
                                03DE 577 ; SAVE THE NAME OF THE COMMUNICATIONS MAILBOX IN THE SPWN BLOCK.
                                03DE 578 ; DEALLOCATE SPACE FOR THE MAILBOX NAME FROM THE SCRATCH STORAGE.
                                03DE 579 ;
14 A6 04 A8 D0 03DE 580 ; MOVL 4(R8),SPWN Q MBXNAM+4(R6) ; SET ADDRESS OF BUFFER
68 10 A6 A2 03E3 581 ; SUBW SPWN Q_MBXNAM(R6),(R8) ; MARK DEVICE NAME NO LONGER SCRATCH

```

```

04 A8 10 A6 C0 03E7 582 ADDL SPWN_Q_MBXNAM(R6),4(R8) ;
03EC 583
03EC 584
03EC 585 :: CREATE AN ATTACH MAILBOX TO HANDLE RE-ATTACH REQUESTS TO THIS PROCESS.
03EC 586
08F3 30 03EC 587 22$: BSBW CREATE_ATTMBX ; CREATE OUR ATTACH MAILBOX
03 50 E8 03EF 588 RO,25$ ; BRANCH IF SUCCESS
02CD 31 03F2 589 95$: BRW SPAWN_EXIT ; BRANCH IF ERROR DETECTED
03F3 590
03F3 591
03F3 592 :: IF /NOTIFY WAS SPECIFIED, CHECK THAT IT IS ALLOWED.
03F3 593
03F3 594 25$: BBC #SPWN V NOTIFY, - ; SKIP IF /NONOTIFY
03F7 595 SPWN_Q_FLAGS(R6),30$
50 00038250 8F D0 03FA 596 MOVL #CLIS NOTIFY,RO ; ASSUME NOTIFY NOT ALLOWED
EC 68 AB 06 E0 0401 597 BBS #PRC V MODE,PRC_W_FLAGS(R11),95$ ; NOT ALLOWED IN BATCH JOBS
02 E0 0406 598 BBS #SPWN V WAIT, - ; NOT ALLOWED IN /WAIT JOBS
E7 0C A6 0408 599 SPWN_Q_FLAGS(R6),95$
040B 600
040B 601
040B 602 :: CHECK THAT COMMAND STRING IS NOT TOO LONG.
040B 603
50 00038018 8F D0 040B 604 30$: MOVL #CLIS BUFOVF,RO ; ASSUME COMMAND BUFFER OVERFLOW
0100 8F 30 A6 B1 0412 605 CMPW SPWN_Q_CMDSTR(R6),#WRK_C_INPBUFS ; COMMAND STRING TOO LONG?
D8 1A 0418 606 BGTRU 95$ ; BRANCH IF SO
041A 607
041A 608
041A 609 :: CHECK THAT PROMPT STRING IS NOT TOO LONG.
041A 610
50 000388FA 8F D0 041A 611 MOVL #CLIS STRTOOLNG,RO ; ASSUME PROMPT IS TOO LONG
00A2 C6 91 0421 612 CMPB SPWN_Q_PROMPTLEN(R6),- ; PROMPT STRING TOO LONG?
23 0425 613 #ENT_K_MAX_PROMPT+3
CA 1A 0426 614 BGTRU 95$ ; BRANCH IF SO
0428 615
0428 616
0428 617 :: PROCESS THE INPUT AND OUTPUT STREAMS.
0428 618
02E5 30 0428 619 BSBW VERIFY_INPUT ; VERIFY INPUT STREAM
C4 50 E9 042B 620 BLBC RO,95$ ; BRANCH IF ERROR
04BE 30 042E 621 BSBW VERIFY_OUTPUT ; VERIFY OUTPUT STREAM
BE 50 E9 0431 622 BLBC RO,95$ ; BRANCH IF ERROR
0434 623
0434 624
0434 625 :: DETERMINE BASE PRIORITY FOR NEW SUBPROCESS FROM CURRENT BASE PRIORITY
0434 626
0434 627
0434 628 CLRO -(SP) ; CREATE GETJPI ITEM LIST
48 A6 DF 0436 628 PUSHAL SPWN_L_PRI8(R6) ; SET BUFFER ADDRESS
03090004 8F DD 0439 629 PUSHL #JPI8_PRI8@16+4 ; REQUEST CURRENT BASE PRIORITY
043F 630
043F 631
043F 632 :: DETERMINE CURRENT IMAGE COUNT (FOR USE BY TERMINATION AST)
043F 633
043F 634
043F 635 CLRL -(SP) ; NO RETLEN ADDRESS
5C A6 DF 0441 635 PUSHAL SPWN_L_IMAGCNT(R6) ; SET BUFFER ADDRESS
041A0004 8F DD 0444 636 PUSHL #JPI8_IMAGECOUNT@16+4 ; REQUEST CURRENT IMAGE COUNT
044A 637
044A 638 :

```

```

044A 639 : DETERMINE CURRENT VALUES OF NON-DEDUCTIBLE QUOTAS, AND CREATE A QUOTA
044A 640 : LIST WHICH WILL BE USED TO SET THE QUOTAS OF THE SUBPROCESS.
044A 641 :
53 60 A6 9E 044A 642 MOVAB SPWN_G_QUOTAS(R6),R3 : GET ADDRESS OF SPACE FOR LIST
7E D4 044E 643 CLRL -(SP) : NO RETLEN ADDRESS
83 01 90 0450 644 MOVB #PQL$ASTLM,(R3)+ : SET TYPE CODE IN PQL LIST
83 DF 0453 645 PUSHAL (R3)+ : SET BUFFER ADDRESS
04090004 8F DD 0455 646 PUSHL #JPI$ASTLM@16+4 : REQUEST CURRENT ASTLM
7E D4 045B 647 CLRL -(SP) : NO RETLEN ADDRESS
83 02 90 045D 648 MOVB #PQL$BIOLM,(R3)+ : SET TYPE CODE IN PQL LIST
83 DF 0460 649 PUSHAL (R3)+ : SET BUFFER ADDRESS
03100004 8F DD 0462 650 PUSHL #JPI$BIOLM@16+4 : REQUEST CURRENT BIOLM
7E D4 0468 651 CLRL -(SP) : NO RETLEN ADDRESS
83 05 90 046A 652 MOVB #PQL$DIOLM,(R3)+ : SET TYPE CODE IN PQL LIST
83 DF 046D 653 PUSHAL (R3)+ : SET BUFFER ADDRESS
03130004 8F DD 046F 654 PUSHL #JPI$DIOLM@16+4 : REQUEST CURRENT DIOLM
7E D4 0475 655 CLRL -(SP) : NO RETLEN ADDRESS
83 0B 90 0477 656 MOVB #PQL$WSDEFAULT,(R3)+ : SET TYPE CODE IN PQL LIST
83 DF 047A 657 PUSHAL (R3)+ : SET BUFFER ADDRESS
04030004 8F DD 047C 658 PUSHL #JPI$DFWSCNT@16+4 : REQUEST CURRENT WSDEFAULT
7E D4 0482 659 CLRL -(SP) : NO RETLEN ADDRESS
83 0A 90 0484 660 MOVB #PQL$WSQUOTA,(R3)+ : SET TYPE CODE IN PQL LIST
83 DF 0487 661 PUSHAL (R3)+ : SET BUFFER ADDRESS
04020004 8F DD 0489 662 PUSHL #JPI$WSQUOTA@16+4 : REQUEST CURRENT WSQUOTA
7E D4 048F 663 CLRL -(SP) : NO RETLEN ADDRESS
83 0D 90 0491 664 MOVB #PQL$WSEXTENT,(R3)+ : SET TYPE CODE IN PQL LIST
83 DF 0494 665 PUSHAL (R3)+ : SET BUFFER ADDRESS
04160004 8F DD 0496 666 PUSHL #JPI$WSEXTENT@16+4 : REQUEST CURRENT WSEXTENT
83 00 90 049C 667 MOVB #PQL$LISTEND,(R3)+ : TERMINATE PQL LIST
50 SE D0 049F 668 MOVL SP,R0 : GET ADDRESS OF ITEM LIST
04A2 669 $GETJPIW S ITMLST=(R0),- : GET CURRENT QUOTA VALUES
04A2 670 IOSB=SPWN_Q_IOSB(R6),-
04A2 671 EFN=#EXEFC_SYSEFN
SE 00000064 8F C0 04BA 672 ADDL #8*12+4,SP : POP ITEM LIST
04C1 673 :
04C1 674 :
04C1 675 : INSERT THE CURRENT SPWN BLOCK INTO THE LINKED LIST OF PENDING SUBPROCESSES,
04C1 676 : SO THAT IN CASE THE SUBPROCESS DIES WHILE WE ARE ATTEMPTING TO FEED IT
04C1 677 : CONTEXT, THE SUBPROCESS TERMINATION AST ROUTINE CAN DO SOMETHING ABOUT IT.
04C1 678 :
04C1 679 :
66 00C0 CB D0 04C6 680 SETBIT SPWN_V_ACTIVE,SPWN_W_FLAGS(R6) : MARK BLOCK CURRENTLY "ACTIVE"
00C0 CB 56 D0 04CB 681 MOVL PRC [ SPWN(R11), SPWN_L_LINK(R6) : INSERT INTO LINKED LIST
04D0 682 MOVL R6,PRC_L_SPWN(R11) : PASS ADDRESS OF SPWN TO TERMINATIO
04D0 683 : AST, SO WE CAN FIND OUT IF IT DIES
04D0 684 :
04D0 685 : CONSTRUCT THE PROCESS NAME OF THE FORM <USERNAME>_1. THE TRAILING
04D0 686 : NUMBER WILL BE INCREMENTED UNTIL WE FIND A UNIQUE PROCESS NAME.
04D0 687 :
04D0 688 :
6F 0C 01 E0 04D0 688 BBS #SPWN_V_PRCNAM,- : BRANCH IF /PROCESS GIVEN
54 01 D0 04D2 689 SPWN_Q_FLAGS(R6),SPAWN_PROCESS : START WITH POSTFIX #1
04D5 690 MOVL #1,R4
04D8 691 :
04D8 692 CONSTRUCT_PRCNAM:
53 18 A6 9E 04D8 693 MOVAB SPWN_Q_PRCNAM(R6),R3 : GET ADDRESS OF DESCRIPTOR SPACE
7E D4 04DC 694 CLRL -(SP) : MARK END OF JPI LIST
53 DD 04DE 695 PUSHL R3 : ADDRESS TO STORE RETURN LENGTH

```

[illegible]

```

18 A6 90 05AC 753 15$: MOVB SPWN_Q_PRCNAM(R6), -      : GET LENGTH OF PROCESS NAME
0092 C6 28 05AF 754      SPWN_T_PROCESS(R6)          :
18 A6 28 05B2 755      MOV C3 SPWN_Q_PRCNAM(R6), -    : COPY THE NAME
1C B6 28 05B3 756      SPWN_Q_PRCNAM+4(R6), -        :
0093 C6 28 05B7 757      SPWN_T_PROCESS+1(R6)         :
05BA 758      :
05BA 759      :
05BA 760      : WRITE /LOG MESSAGE INDICATING PROCESS WAS CREATED
05BA 761      :
00 00 00 05BA 762      BBC #SPWN_V_LOG, -             : BRANCH IF /NOLOG SPECIFIED
10 0C A6 E1 05BC 763      SPWN_Q_FLAGS(R6), 20$        :
18 A6 9F 05BF 764      PUSHAB SPWN_Q_PRCNAM(R6)        : ADDRESS OF PROCESS NAME
51 01 D0 05C2 765      MOVL #1, RT                     : SET NUMBER OF ARGS
50 0003FD01 8F D0 05C3 766      MOVL #CLIS SPAWNED, R0   : MESSAGE CODE
FA31 30 05CC 767      BSBW DCL$FORMMSG                 : OUTPUT MESSAGE
05CF 768      :
05CF 769      :
05CF 770      : IF /WAIT /LOG THEN WRITE MESSAGE INDICATING SUBPROCESS ATTACHED.
05CF 771      : DO THIS BEFORE WE WRITE THE CONTEXT, WHILE WE STILL HAVE CONTROL OVER
05CF 772      : THE TERMINAL.
05CF 773      :
02 02 E1 05CF 774 20$: BBC #SPWN_V_WAIT, -             : SKIP IF /NOWAIT
15 0C A6 E1 05D1 775      SPWN_Q_FLAGS(R6), 30$        :
00 00 E1 05D4 776      BBC #SPWN_V_LOG, -             : SKIP IF /NOLOG SPECIFIED
10 0C A6 9F 05D6 777      SPWN_Q_FLAGS(R6), 30$        :
18 A6 9F 05D9 778      PUSHAB SPWN_Q_PRCNAM(R6)        : ADDRESS OF PROCESS NAME
51 01 D0 05DC 779      MOVL #1, RT                     : SET NUMBER OF ARGS
50 0003FD09 8F D0 05DF 780      MOVL #CLIS ATTACHED, R0  : MESSAGE CODE
FA17 30 05E6 781      BSBW DCL$FORMMSG                 : OUTPUT MESSAGE
05E9 782      :
05E9 783      :
05E9 784      : WRITE CONTEXT TO SUBPROCESS VIA MAILBOX. IF AN ERROR OCCURS AT THIS POINT,
05E9 785      : WE MUST DELETE THE PROCESS TO AVOID HAVING IT HANG AROUND FOR NOTHING.
05E9 786      :
23 0C A6 0E E0 05E9 787 30$: BBS #SPWN_V_NOCTX, SPWN_W_FLAGS(R6), 35$ : BRANCH IF UNKNOWN CLI WAS SPECIF
088F 30 05EE 788      BSBW WRITE_CONTEXT              : WRITE CONTEXT TO SUBPROCESS
1D 50 E8 05F1 789      BLBS R0, 35$                   : BRANCH IF OK
50 DD 05F4 790      PUSHL R0                          : SAVE WRITE STATUS
05F6 791      $DELPRC_S PIDADR=SPWN_L_SUBPID(R6)      : DELETE THE SUBPROCESS
01 44 A6 D1 0602 792      CMPL SPWN_L_STATUS(R6), #1   : GET TERMINATION STATUS
04 1B 0606 793      BLEQU 32$                          : USE IT IF SIGNIFICANT
6E 44 A6 D0 0608 794      MOVL SPWN_L_STATUS(R6), (SP) : (NEQ TO 0 OR 1)
50 8ED0 060C 795 32$: POPL R0                         : RESTORE STATUS CODE
6B 11 060F 796      BRB 60$                          : SIGNAL THE ORIGINAL ERROR, OR THE
0611 797      : SUBPROCESS ABNORMAL STATUS (IF ANY)
0611 798      :
0611 799      :
0611 800      : CLEANUP CONTEXT MAILBOX, TO ELIMINATE POOL USAGE AS SOON AS POSSIBLE.
0611 801      :
0611 802 35$: $DASSGN_S CHAN=SPWN_W_CHAN(R6)          : CLEANUP MAILBOX (THUS, POOL USAGE)
0A A6 B4 061C 803      CLRW SPWN_W_CHAN(R6)           : INDICATE CHANNEL WAS "REMOVED"
50 01 D0 061F 804      MOVL #1, R0                    : IGNORE STATUS FROM DEASSIGN
0622 805      : WHILE WAITING FOR RE-ATTACH
0622 806      :
0622 807      :
0622 808      : CREATE /NOWAIT SUBPROCESS.
0622 809      :

```

```

51 01 D0 0622 810      MOVL  #1,R1      : SET NORMAL TERMINATION STATUS
      02 E0 0623 811      BBS   #SPWN_V_WAIT,-      : BRANCH IF /NOWAIT
03 0C A6 0627 812      BRW   SPWN_Q_FLAGS(R6),9000$ ;
      0095 31 062A 813      :
      062D 814 9000$:
      062D 815 :
      062D 816 : CREATE /WAIT SUBPROCESS. MARK THIS (PARENT) PROCESS DETACHED FROM TERMINAL.
      062D 817 :
      07 E1 062D 818      BBC   #SPWN_V_ACTIVE,-      : BRANCH IF INACTIVE (SUBPROCESS
4A 0C A6 062F 819      SETBIT SPWN_Q_FLAGS(R6),60$      : ALREADY TERMINATED)
      0632 820      SETBIT PRC_V_DETACHED,PRC_W_FLAGS(R11) : MARK PROCESS DETACHED
      0637 821 :
      0637 822 :
      0637 823 : WAIT FOR SUBPROCESS TO RETURN CONTROL HERE (VIA ATTACH OR TERMINATION)
      0637 824 :
      0883 30 0637 825 50$: BSBW  CHECK_FOR_HANGUP      : CHECK FOR HANGUP AST
50 7E 7E 063A 826      MOVAQ  -(SP),R0      : ALLOCATE A TEMP. IOSB
      063D 827      $QIOW_S FUNC=#IOS SETMODE!IOSM_WRTATTN,- : SET ATTENTION AST ON MAILBOX
      063D 828      CHAN=PRC_Q_ATTMBX(R11),-
      063D 829      EFN=#EXE$C_SYSEFN,-
      063D 830      IOSB=(R0),-
      063D 831      P1=ATTACH_AST,-
      063D 832      P2=R11
SE 08 C0 0663 833      ADDL   #8,SP      : ADDRESS OF AST ROUTINE
      0666 834      $HIBER_S : PASS ADDRESS OF CLI STORAGE
      07 E1 066D 835      BBC   #SPWN_V_ACTIVE,-      : CLEAN UP STACK
05 0C A6 066F 836      BBS   SPWN_Q_FLAGS(R6),55$      : HIBERNATE WAITING FOR WAKEUP
      0F E0 0672 837      BBS   #PRC_V_DETACHED,-      : BRANCH IF SUBPROCESS INACTIVE
C0 68 AB 0674 838      BBS   PRC_Q_FLAGS(R11),50$      : (SUBPROCESS HAS BEEN TERMINATED)
      0677 839 55$: CLRBIT PRC_V_DETACHED,PRC_W_FLAGS(R11) : BRANCH IF PARENT STILL DETACHED
      067C 840 : MARK PROCESS ATTACHED TO TERMINAL
      067C 841 :
      067C 842 : WE HAVE RETURNED FROM THE SUBPROCESS EITHER VIA A RE-ATTACH REQUEST TO
      067C 843 : THIS PROCESS OR VIA SUBPROCESS TERMINATION. IF /LOG WAS SPECIFIED, THEN
      067C 844 : OUTPUT THE RETURNED MESSAGE.
      067C 845 :
      00 E1 067C 846 60$: BBC   #SPWN_V_LOG,-      : BRANCH IF /NOLOG SPECIFIED
08 0C A6 067E 847      PUSHL  R0      :
      50 DD 0681 848      BSBW  RETURNED_MESSAGE      : SAVE FINAL STATUS
      07B0 30 0683 849      POPL  R0      : WRITE MESSAGE SAYING WE HAVE RETUR
      50 8ED0 0686 850 : RESTORE FINAL STATUS
      0689 851 :
      0689 852 :
      0689 853 : IF WE ARE IN AN INTERACTIVE PROCESS, THEN TELL THE TERMINAL DRIVER THAT
      0689 854 : WE NOW OWN THE TERMINAL.
      0689 855 :
      28 68 AB 06  E0 0689 856 70$: BBS   #PRC_V_MODE,PRC_W_FLAGS(R11),80$ : SKIP IF NOT INTERACTIVE
      50 DD 068E 857      PUSHL  R0      : SAVE FINAL STATUS
      0690 858      $QIOW_S FUNC=#IOS SETMODE!IOSM TT_PROCESS,- : ASSUME TERMINAL OWNERSHIP
      0690 859      CHAN=PRC_Q_INPCHAN(R11),-
      0690 860      IOSB=SPWN_Q_IOSB(R6),-
      0690 861      EFN=#EXE$C_SYSEFN
      50 8ED0 0683 862      POPL  R0      : RESTORE FINAL STATUS
      0686 863 :
      0686 864 :
      0686 865 : IF THIS IS A RE-ATTACH THEN SKIP DATA STRUCTURE AND MAILBOX DELETIONS AND
      0686 866 : SET NORMAL TERMINATION STATUS. OTHERWISE, SET THE RETURNED TERMINATION STATUS.

```

```

51 01 DO 06B6 867 :
      07 EO 06B6 868 80$: MOVL #1,R1 : SET NORMAL TERMINATION STATUS
33 0C A6 06B9 869 : BBS #SPWN_V_ACTIVE,- : BRANCH IF CONTROL RETURNED
51 44 A6 DO 06BB 870 : SPWN_Q_FLAGS(R6),RESTORE_CONTEXT: VIA A RE-ATTACH
      06BE 871 : MOVL SPWN_L_STATUS(R6),R1 : GET TERMINATION STATUS
      06C2 872 :
      06C2 873 :
      06C2 874 : THE SPAWN COMMAND IS ESSENTIALLY COMPLETE. WE HAVE RETURNED CONTROL HERE
      06C2 875 : DUE TO SUBPROCESS TERMINATION. THEREFORE, CLEANUP MISC. DATA STRUCTURES AND
      06C2 876 : MAILBOXES AND REPORT ANY ERRORS WHICH WERE DETECTED.
      06C2 877 :
      06C2 878 : IF SUBPROCESS WAS SUCCESSFULLY SPAWNED AND TERMINATED, THEN TMBX WAS
      06C2 879 : DELETED BY THE TERMINATION AST. IF NOT, TRY TO DEASSIGN THE MAILBOX AND
      06C2 880 : DELETE THE DATA STRUCTURE HERE.
      06C2 881 :
      06C2 882 SPAWN_EXIT:
      06C2 883 : MOVQ R0,-(SP) : SAVE ERROR AND TERMINATION STATUS
      06C5 884 : BLBS R0,10$ : BRANCH IF SUCCESSFUL SPAWN
      06C8 885 : BBC #SPWN_V_ACTIVE,- : DID SUBPROCESS TERMINATE?
      06CA 886 : SPWN_Q_FLAGS(R6),10$ : IF SO, TERMINATION MBX TAKEN CARE
      06CD 887 : DECB TMBX_B_REFS(R7) : DECR REFERENCE COUNT TO MAILBOX
      06D0 888 : BNEQ 10$ : BRANCH IF STILL OUTSTANDING USES
      06D2 889 : BSBW DELETE_TMBX : DELETE TERMINATION MAILBOX AND TMB
      06D5 890 :
      06D5 891 :
      06D5 892 : IN ALL CASES, DELETE THE ATTACH MAILBOX AND DEASSIGN THE CONTEXT MAILBOX
      06D5 893 : CHANNEL.
      06D5 894 :
      06D5 895 10$: BSBW DELETE_ATTMBX : DELETE THE ATTACH MAILBOX
      06D8 896 : $DASSGN_S_CHAN=SPWN_W_CHAN(R6) : DEASSIGN CONTEXT MAILBOX CHANNEL
      06E3 897 : CLRW SPWN_W_CHAN(R6) : INDICATE CHANNEL WAS 'REMOVED'
      06E6 898 :
      06E6 899 :
      06E6 900 : IF /WAIT, THEN DEALLOCATE THE SPWN BLOCK.
      06E6 901 :
      06E6 902 : BBC #SPWN_V_WAIT,- : BRANCH IF /NOWAIT
      06E8 903 : SPWN_Q_FLAGS(R6),20$ :
      06EB 904 : BSBW DEALLOC_SPWN : REMOVE INACTIVE SPWN BLOCKS
      06EE 905 20$: MOVQ (SP)+,R0 : RESTORE ERROR AND TERMINATION STAT
      06F1 906 :
      06F1 907 :
      06F1 908 : IN ALL CASES, RESTORE THE PROCESS STATE -
      06F1 909 : STACK, EVENT FLAGS, CONTROL KEY AST'S, SPAWN STATUS, TERMINATION STATUS,
      06F1 910 : AND SAVED REGISTERS.
      06F1 911 :
      06F1 912 RESTORE_CONTEXT:
      06F1 913 : MOVAB 768+8(SP),SP : POP SCRATCH STORAGE OFF STACK
      06F6 914 : MOVQ R0,-(SP) : SAVE ERROR AND TERMINATION STATUS
      06F9 915 : MOVL SPWN_L_OUTOFBAND(R6),R1 : RESTORE OUT-OF-BAND ENABLE MASK
      06FD 916 : BSBW DCL$RESET00B : RESTORE AST'S
      0700 917 : POPR #^M<R0,R1,R2,R3,R4,R5,R7,R8> : RESTORE REGISTERS
      0704 918 :
      0704 919 :
      0704 920 : IF A HANGUP IS PENDING, THEN ABORT PROCESS
      0704 921 :
      0704 922 : BBC #PRC_V_HANGUP,- : IF SET, HANG-UP PENDING
      0706 923 : PRC_Q_FLAGS(R11),20$ :

```

SPAWN
V04-000

- MULTI-PROCESSING COMMANDS
SPAWN A SUBPROCESS

D 13

16-SEP-1984 00:17:05
4-SEP-1984 23:43:20

VAX/VMS Macro V04-00
[DCL.SRC]SPAWN.MAR;1

Page 19
(4)

SE 08 C0 0709 924 ADDL
F8F1 31 070C 925 BRW
05 070F 926 208: RSB

#8.SP
CCL\$ABORT

: POP TWO PC'S OFF THE STACK
: LOG THE PROCESS OUT
: EXIT

```

0710 928 .SBTTL PROCESS SPAWN INPUT STREAM
0710 929
0710 930
0710 931
0710 932
0710 933
0710 934
0710 935
0710 936
0710 937
0710 938
0710 939
0710 940
0710 941
0710 942
0710 943
0710 944
0710 945
0710 946
0710 947
0710 948
0710 949
0710 950
0710 951
0710 952
0710 953
0710 954
0710 955
0710 956
0710 957
0710 958
0710 959
0710 960
0710 961
0710 962
0710 963
0710 964
0710 965
0710 966
0710 967
0710 968
0710 969
0710 970
0710 971
0710 972
0710 973
0710 974
0710 975
0710 976
0710 977
0710 978
0710 979
0710 980
0710 981
0710 982
0710 983
0710 984

    20 A6 15 0C 0A E0
    24 A6 F921 CF 9A
    30 A6 F91C CF 9E
    04 B5 0721 956
    13 0724 957
    0726 958
    0726 959
    072A 960
    072A 961
    072A 962
    072A 963
    50 00038200 8F D0
    20 A6 B1
    00FF 8F B1
    5C 1A

    5E FF50 CE 9E
    00 6E 00 2C
    6E 0080 8F
    54 5E D0
    55 50 A4 9E

    5003 8F B0
    64

    34 A4 20 A6 90
    2C A4 24 A6 D0
    35 A4 F8FB CF 90

    108:
    MOVL #CLIS_INVFILSPE,R0
    CMPW SPWN_Q_INPUT(R6),-
    BGTRU #NAMSC_MAXRSS

    108:
    MOVAB -FABSC_BLN-NAMSC_BLN(SP),SP
    MOVCS #0,(SP),#0,-
    #FABSC_BLN+NAMSC_BLN,(SP)
    MOVL SP,R4
    MOVAB FABSC_BLN(R4),R5

    ASSUME FABSB_BLN EQ FABSB_BID+1
    MOVW #FABSC_BID+<FABSC_BLN28>,-
    FABSB_BID(R4)
    SETBIT FABSV_PPF,FABSL_FOP(R4)
    MOVAB SPWN_Q_INPUT(R6),FABSB_FNS(R4)
    MOVL SPWN_Q_INPUT+4(R6),FABSL_FNA(R4)
    MOVAB COM,FABSB_DNS(R4)

    IF INPUT SPECIFIED,
    KEEP AUTOLOGUT FLAG OFF
    USE "SYSSINPUT"
    ANY COMMAND STRING SPECIFIED?
    NO, DO NOT SET IMPLIED LOGOUT
    YES, SET IMPLIED LOGOUT FLAG

    CHECK THAT INPUT FILE SPECIFICATION IS NOT TOO LONG.
    ASSUME INVALID FILE SPEC
    NAME TOO LONG?
    IF SO, EXIT WITH ERROR

    ALLOCATE FAB AND NAM BLOCKS AND THEN SPARSE THE INPUT FILE SPECIFICATION.
    ALLOCATE FAB AND NAM BLOCKS
    ZERO FAB AND NAM
    GET BASE OF FAB
    GET BASE OF NAM
    INIT FAB
    SET FAB ID
    USE PROCESS I/O SEGMENT
    SPECIFY FILE NAME
    SPECIFY DEFAULT FILE TYPE

```

```

30 A4 F8F6 CF 9E 0767 985 MOVAB COM+1,FAB$L_DNA(R4)
28 A4 55 D0 076D 986 MOVL R5,FAB$L_NAM(R4)
0771 987
0771 988 ASSUME NAM$B_BLN EQ NAM$B_BID+1
6002 8F B0 0771 989 MOVW #NAM$C_BID+<NAM$C_BLN$B>,-
65 0775 990 NAM$B_BID(R5)
08 A5 10 90 0776 991 MOVW #NAM$B_NOCONCEAL,NAM$B_NOP(R5)
0A A5 FF 8F 90 077A 992 MOVW #NAM$C_MAXRSS,NAM$B_ESS(R5)
0C A5 04 AB D0 077F 993 MOVL 4(R8),NAM$B_E$A(R5)
0784 994
0784 995 SPARSE FAB=(R4)
08 50 E8 078D 996 BLBS R0,30$
0790 997
5E 00B0 CE 9E 0790 998 20$: MOVAB FAB$C_BLN+NAM$C_BLN(SP),SP
00EB 31 0795 999 25$: BRW 95$
0798 1000
0798 1001
0798 1002
0798 1003
0798 1004
0798 1005
0798 1006 30$: BBS #NAM$V_PPF,NAM$B_FNB(R5),50$
50 0B A5 9A 079D 1007 MOVZBL NAM$B_ESL(R5),R0
68 50 C2 07A1 1008 SUBL R0,(R8)
04 A8 50 C0 07A4 1009 ADDL R0,4(R8)
0D 40 A4 00 E1 07A8 1010 BBC #DEV$V_REC,FAB$B_DEV(R4),40$
20 A6 39 A5 9A 07AD 1011 MOVZBL NAM$B_DEV(R5),SPWN_Q_INPUT(R6)
24 A6 44 A5 D0 07B2 1012 MOVL NAM$B_DEV(R5),SPWN_Q_INPUT+4(R6)
00B5 31 07B7 1013 BRW 56$
07BA 1014
07BA 1015
07BA 1016
07BA 1017
07BA 1018
07BA 1019
07BA 1020 40$: $SEARCH FAB=(R4)
CA 50 E9 07C3 1021 BLBC R0,20$
20 A6 0B A5 9A 07C6 1022 MOVZBL NAM$B_ESL(R5),SPWN_Q_INPUT(R6)
24 A6 0C A5 D0 07CB 1023 MOVL NAM$B_E$A(R5),SPWN_Q_INPUT+4(R6)
07D0 1024 SETBIT SPWN_Q_MODE,SPWN_W_FLAGS(R6)
5E 00B0 CE 9E 07D4 1025 45$: MOVAB FAB$C_BLN+NAM$C_BLN(SP),SP
0095 31 07D9 1026 BRW 90$
07DC 1027
07DC 1028
07DC 1029
07DC 1030
07DC 1031
07DC 1032
07DC 1033
07DC 1034 50$: BBC #SPWN_V_INPUT,SPWN_W_FLAGS(R6),52$
0C 0C A6 0A E1 07DC 1035 MOVL #CLIS-SPWNIO,R0
50 000388EA 8F D0 07E1 1036 BBC #DEV$V_REC,FAB$B_DEV(R4),20$
A3 40 A4 00 E1 07E8 1037
07ED 1038
07ED 1039
07ED 1040
07ED 1041 52$: BBS #DEV$V_REC,FAB$B_DEV(R4),56$

```

: SPECIFY NAM BLOCK ADDRESS
 : INIT NAM
 : SET NAM ID
 : GET PAST DOUBLE UNDERSCORES
 : SPECIFY RESULT BUFFER
 : PARSE THE FILE SPEC
 : CONTINUE IF SUCCESSFUL
 : DEALLOCATE FAB AND NAM
 : EXIT WITH ERROR
 : PROCESS NON-PPF INPUT HERE. IF NOT RECORD-ORIENTED, THEN USE EXPANDED
 : STRING AS INPUT, DO A \$SEARCH FOR THE FILE, AND SET SILENT LOGOUT FLAG.
 : IF RECORD-ORIENTED THEN USE DEV FIELD AS INPUT.
 : BRANCH IF PPF FILE
 : GET LENGTH OF RESULT STRING
 : PERMANENTLY REMOVE RESULT
 : FROM SCRATCH AREA
 : BRANCH IF NOT REC-ORIENTED
 : USE DEVICE AS INPUT
 : PROCESS AS PPF DEVICE
 : INPUT IS A NON-PPF NON-RECORD-ORIENTED DEVICE. \$SEARCH FOR THE FILE.
 : IF FOUND, THEN USE THE ESS AS SYSSINPUT. ALSO, MARK THE PROCESS AS
 : NON-INTERACTIVE.
 : SEARCH FOR THE SPECIFIED FILE
 : BRANCH IF NOT FOUND
 : USE ESS AS INPUT FILE SPEC
 : SET NON-INTERACTIVE SUBPROCESS
 : DEALLOCATE FAB AND NAM
 : DONE WITH PARSING
 : INPUT IS PPF. IF FROM A RECORD-ORIENTED DEVICE, DO \$GETDVI TO GET
 : THE DEVICE NAME. OTHERWISE, 1) IF EXPLICITLY SPECIFIED THEN RETURN AN
 : ERROR, 2) IF INTERACTIVE PROCESS THEN USE THE TERMINAL AS INPUT, 3) IF
 : NON-INTERACTIVE PROCESS THEN USE NL: AS INPUT.
 : SKIP IF IMPLICIT INPUT
 : ASSUME ILLEGAL INPUT FILE
 : ERROR IF NOT REC-ORIENTED
 : PROCESS LEGAL NON-RECORD-ORIENTED PPF INPUT.
 : BRANCH IF REC-ORIENTED

```

37 68 AB 06 E0 07F2 1042 BBS #PRC V MODE,PRC W FLAGS(R11),53$ : BRANCH IF NOT INTERACTIVE
51 00000028 8F D0 07F7 1043 MOVL #CTLSAG CLIDATA+PPD$T_INPDV1,R1 : GET ADDRESS OF ASCII DEVICE NAME
55 81 9A 07FE 1044 MOVZBL (R1)+,R5 : GET LENGTH OF DEVICE
7E 54 7D 0801 1045 MOVQ R4,-(SP) : SAVE R4/R5
04 88 61 55 28 0804 1046 MOVQ R5,(R1),R4(R8) : COPY STRING TO SCRATCH AREA
63 3A 90 0809 1047 MOVQ #A',,(R3) : ADD COLON TO END OF STRING
54 8E 7D 080C 1048 MOVQ (SP)+,R4 : RESTORE R4/R5
55 D6 080F 1049 INCL R5 : ADD LENGTH OF A COLON
20 A6 55 D0 0811 1050 MOVL R5,SPWN_Q_INPUT(R6) : SAVE DEVICE DESCRIPTOR
24 A6 04 A8 D0 0815 1051 MOVL 4(R8),SPWN_Q_INPUT+4(R6)
68 55 A2 081A 1052 SUBW R5,(R8) : PERMANENTLY REMOVE DEVICE
04 A8 55 C0 081D 1053 ADDL R5,4(R8) : FROM SCRATCH AREA
00000044 9F D0 0821 1054 MOVL #CTLSAG CLIDATA+PPD$S_INPDEV,- : GET INPUT DEVICE CHARS
40 A4 0827 1055 FAB$S_DEV(R4)
26 40 A4 00 E0 0829 1056 BBS #DEV$Q_REC,FAB$S_DEV(R4),58$ : BRANCH IF REC-ORIENTED
20 A6 F829 CF 9A 082E 1057 53$: MOVZBL NL,SPWN_Q_INPUT(R6) : SET INPUT TO NL:
24 A6 F824 CF 9E 0834 1058 MOVAB NL+1,SPWN_Q_INPUT+4(R6)
40 A4 D4 083A 1059 CLRL FAB$S_DEV(R4) : CLEAR DEVICE CHARACTERISTICS
15 11 083D 1060 BRB 58$
083F 1061
083F 1062
083F 1063 : PROCESS REC-ORIENTED PPF INPUT.
083F 1064
52 20 A6 7D 083F 1065 56$: MOVQ SPWN_Q_INPUT(R6),R2 : COPY DESCRIPTOR OF DEVICE NAME
54 68 7D 0843 1066 MOVQ (R8),R4 : COPY DESCRIPTOR OF SCRATCH AREA
097D 30 0846 1067 BSBW GET_DEVICE : GET DEVICE NAME
20 A6 54 7D 0849 1068 MOVQ R4,SPWN_Q_INPUT(R6) : SAVE EQUIVALENCE STRING
68 54 A2 084D 1069 SUBW R4,(R8) : PERMANENTLY REMOVE RESULT
04 A8 54 C0 0850 1070 ADDL R4,4(R8) : FROM SCRATCH AREA
0854 1071
0854 1072 : GET TERMINAL CHARACTERISTICS. DO NOT SPAWN THE SUBPROCESS IF THE TERMINAL
0854 1073 : HAS AN ASSOCIATED MAILBOX.
0854 1074
0854 1075
50 40 AE D0 0854 1076 58$: MOVL FAB$S_DEV(SP),R0 : GET DEVICE CHARACTERISTICS
5E 0080 CE 9E 0858 1077 MOVAB FAB$C_BLN+NAM$C_BLN(SP),SP : DEALLOCATE FAB AND NAM
06 50 02 E0 085D 1078 BBS #DEV$Q_TRM,R0,59$ : SKIP IF INPUT IS NOT A TERMINAL
0A 11 0861 1079 SETBIT SPWN_V_MODE,SPWN_W_FLAGS(R6) : SET NON-INTERACTIVE SUBPROCESS
51 20 A6 7E 0865 1080 BRB 90$ : ALL DONE
0016 30 0867 1081 59$: MOVAQ SPWN_Q_INPUT(R6),R1 : GET ADDRESS OF INPUT STRING
12 50 E9 0868 1082 BSBW CHECK_TRMBX : CHECK FOR MAILBOX
086E 1083 BLBC R0,95$ : BRANCH IF IT EXISTS
0871 1084
0871 1085 : CHECK THAT INPUT FILE SPEC IS NOT TOO LONG FOR SCREPRC TO HANDLE.
0871 1086
0871 1087
50 00038200 8F D0 0871 1088 90$: MOVL #CLIS_INVFILSPE,R0 : ASSUME SPEC IS TOO LONG
20 A6 00FF 8F B1 0878 1089 CMPW #LNMSC_NAMLENGTH,SPWN_Q_INPUT(R6) : IS IT REALLY?
03 1F 087E 1090 BLSSU 95$ : YES, THEN ERROR
0880 1091
0880 1092 : RETURN WITH STATUS
0880 1093
0880 1094
50 01 D0 0880 1095 : SET NORMAL STATUS
05 0883 1096 95$: RSB : EXIT WITH STATUS

```

```

: SAVE REGISTERS
: ALLOCATE AN IOSB
: ALLOCATE LONGWORD FOR CHANNEL NUMB
: ALLOCATE DEVICE CHARACTERISTICS BU
: SAVE BUFFER ADDRESS
: GET A CHANNEL TO THE TERMINAL
:
: BRANCH IF ERROR
:
: GET TERMINAL CHARACTERISTICS
:
:
:
: BRANCH IF ERROR
: GET IOSB STATUS
: BRANCH IF ERROR
:
: DEASSIGN TERMINAL CHANNEL
: BRANCH IF ERROR
:
: ASSUME ASSOCIATED TERMINAL MBX
: EXIT WITH ERROR IF TRUE
: SET SUCCESS
:
: RESTORE STACK
: RESTORE REGISTERS
:

```

```

08EF 1145 .SBTTL PROCESS SPAWN OUTPUT STREAM
08EF 1146 :---
08EF 1147 :
08EF 1148 THIS ROUTINE IS CALLED TO PROCESS THE SPAWN OUTPUT STREAM.
08EF 1149 :
08EF 1150 INPUTS:
08EF 1151 :
08EF 1152 R6 = ADDRESS OF SPAWN BLOCK
08EF 1153 R8 = DESCRIPTOR OF SCRATCH AREA
08EF 1154 R11 = ADDRESS OF PRC AREA
08EF 1155 :
08EF 1156 OUTPUTS:
08EF 1157 :
08EF 1158 R0 = STATUS CODE
08EF 1159 :---
08EF 1160 :
08EF 1161 VERIFY_OUTPUT:
08EF 1162 :
08EF 1163 IF OUTPUT FILE SPECIFICATION NOT SPECIFIED, THEN USE CURRENT SYSS$OUTPUT
08EF 1164 :
08EF 1165 BBS #SPWN_V_OUTPUT, - ; IF OUTPUT SPECIFIED,
08F1 1166 SPWN_Q_FLAGS(R6), 10$ ; THEN USE IT
28 A6 F74C CF 9A 08F4 1167 MOVZBL SYSS$OUTPUT, SPWN_Q_OUTPUT(R6) ; ELSE USE "SYSS$OUTPUT"
2C A6 F747 CF 9E 08FA 1168 MOVAB SYSS$OUTPUT+1, SPWN_Q_OUTPUT+4(R6);
0900 1169 :
0900 1170 :
0900 1171 CHECK THAT OUTPUT FILE SPECIFICATION IS NOT TOO LONG.
0900 1172 :
50 00038200 8F D0 0900 1173 10$: MOVL #CLIS_INVFILSPE, R0 ; ASSUME INVALID FILE SPEC
00FF 8F 28 A6 B1 0907 1174 CMPW SPWN_Q_OUTPUT(R6), #NAM$C_MAXRSS ; NAME TOO LONG?
5C 1A 090D 1175 BGTRU 25$ ; IF SO, EXIT WITH ERROR
090F 1176 :
090F 1177 :
090F 1178 ALLOCATE FAB AND NAM BLOCKS AND THEN SPARSE THE OUTPUT FILE SPECIFICATION.
090F 1179 :
5E FF50 CE 9E 090F 1180 MOVAB -FAB$C_BLN-NAM$C_BLN(SP), SP ; ALLOCATE FAB AND NAM BLOCKS
00 6E 00 2C 0914 1181 MOVCS #0, (SPT, #0, - ; ZERO FAB AND NAM
6E 0080 8F 0918 1182 #FAB$C_BLN+NAM$C_BLN, (SP)
54 5E D0 091C 1183 MOVL SP, R4 ; GET BASE OF FAB
55 50 A4 9E 091F 1184 MOVAB FAB$C_BLN(R4), R5 ; GET BASE OF NAM
0923 1185 :
0923 1186 ASSUME FAB$B_BLN EQ FAB$B_BID+1 ; INIT FAB
0923 1187 MOVW #FAB$C_BID+<FAB$C_BLN$B>, - ; SET FAB ID
0927 1188 FAB$B_BID(R4)
0928 1189 SETBIT FAB$V_PPF, FAB$L_FOP(R4) ; USE PROCESS I/O SEGMENT
34 A4 28 A6 90 092D 1190 MOVB SPWN_Q_OUTPUT(R8), FAB$B_FNS(R4) ; SPECIFY FILE NAME
2C A4 2C A6 D0 0932 1191 MOVL SPWN_Q_OUTPUT+4(R6), FAB$L_FNA(R4);
35 A4 F72A CF 90 0937 1192 MOVB LOG, FAB$B_DNS(R4) ; SPECIFY DEFAULT FILE TYPE
30 A4 F725 CF 9E 093D 1193 MOVAB LOG+1, FAB$L_DNA(R4)
28 A4 55 D0 0943 1194 MOVL R5, FAB$L_NAM(R4) ; SPECIFY NAM BLOCK ADDRESS
0947 1195 :
0947 1196 ASSUME NAM$B_BLN EQ NAM$B_BID+1 ; INIT NAM
0947 1197 MOVW #NAM$C_BID+<NAM$C_BLN$B>, - ; SET NAM ID
094B 1198 NAM$B_BID(R5)
08 A5 10 90 094C 1199 MOVB #NAM$B_NOCONCEAL, NAM$B_NOP(R5) ; GET PAST DOUBLE UNDERSCORES
0A A5 FF 8F 90 0950 1200 MOVB #NAM$C_MAXRSS, NAM$B_ESS(R5) ; SPECIFY RESULT BUFFER
0C A5 04 A8 D0 0955 1201 MOVL 4(R8), NAM$B_ESA(R5)

```

```

08 50 E8 095A 1202 SPARSE FAB=(R4) : PARSE THE FILE SPEC
0963 1203 BLBS R0,308 : CONTINUE IF SUCCESSFUL
0966 1204
SE 00B0 CE 9E 0966 1205 20$: MOVAB FAB$C_BLN+NAME$C_BLN(SP),SP : DEALLOCATE FAB AND NAME
0092 31 096B 1206 25$: BRW 95$ : EXIT WITH ERROR
096E 1207
096E 1208
096E 1209 : PROCESS NON-PPF OUTPUT HERE. IF NOT RECORD-ORIENTED, THEN USE EXPANDED
096E 1210 : STRING AS OUTPUT. IF RECORD-ORIENTED THEN USE DEV FIELD AS OUTPUT.
096E 1211
28 34 A5 10 E0 096E 1212 30$: BBS #NAME$V_PPF,NAME$C_FNB(R5),50$ : BRANCH IF PPF FILE
50 0B A5 9A 0973 1213 MOVZBL NAME$B_ESL(R5),R0 : GET LENGTH OF RESULT STRING
68 50 C2 0977 1214 SUBL R0,(R8) : PERMANENTLY REMOVE RESULT
04 A8 50 C0 097A 1215 ADDL R0,4(R8) : FROM SCRATCH AREA
0C 40 A4 00 E1 097E 1216 #DEV$V_REC,FAB$C_DEV(R4),40$ : BRANCH IF NOT REC-ORIENTED
28 A6 39 A5 9A 0983 1217 MOVZBL NAME$B_DEV(R5),SPWN_Q_OUTPUT(R6) : USE DEVICE AS OUTPUT
2C A6 44 A5 D0 0988 1218 MOVL NAME$C_DEV(R5),SPWN_Q_OUTPUT+4(R6)
40 11 098D 1219 BRB 56$ : PROCESS AS PPF DEVICE
098F 1220
098F 1221
098F 1222 : OUTPUT IS A NON-PPF NON-RECORD-ORIENTED DEVICE. USE THE ESS AS OUTPUT.
098F 1223
28 A6 0B A5 9A 098F 1224 40$: MOVZBL NAME$B_ESL(R5),SPWN_Q_OUTPUT(R6) : USE ESS AS OUTPUT FILE SPEC
2C A6 0C A5 D0 0994 1225 MOVL NAME$C_ESA(R5),SPWN_Q_OUTPUT+4(R6)
49 11 0999 1226 BRB 90$ : DONE WITH PARSING
099B 1227
099B 1228
099B 1229 : OUTPUT IS PPF. IF FROM A RECORD-ORIENTED DEVICE, DO $GETDVI TO GET
099B 1230 : THE DEVICE NAME. OTHERWISE, IF EXPLICITLY SPECIFIED THEN RETURN AN
099B 1231 : ERROR ELSE CREATE A MAILBOX THAT WILL CAUSE WRITES TO GO TO THE PARENT'S
099B 1232 : OUTPUT STREAM.
099B 1233
0E 0C A6 0B E1 099B 1234 50$: BBC #SPWN_V_OUTPUT,SPWN_W_FLAGS(R6),51$ : SKIP IF IMPLICIT OUTPUT
50 000388EA 8F D0 09A0 1235 MOVL #CLIS_SPWNIO,R0 : ASSUME ILLEGAL OUTPUT FILE
BA 40 A4 00 E1 09A7 1236 BBC #DEV$V_REC,FAB$C_DEV(R4),20$ : ERROR IF NOT REC-ORIENTED
17 11 09AC 1237 BRB 52$
09AE 1238
09AE 1239
09AE 1240 : IF AT CTRL/Y LEVEL, THEN DON'T USE SYS$OUTPUT. USE THE OUTPUT STREAM
09AE 1241 : SPECIFIED AT PROCESS CREATION TIME.
09AE 1242
12 68 0B E1 09AE 1243 51$: BBC #PRC_V_YLEVEL - : IF CTRL/Y LEVEL
0118 AB 09B0 1244 PRC_Q_FLAGS(R11),52$ : THEN SPECIAL CASE SYS$OUTPUT
40 CB D0 09B3 1245 MOVL PRC_L_OUTRABCTX(R11),- : GET DEVICE CHARACTERISTICS
51 011C CB 9E 09B7 1246 FAB$C_DEV(R4)
50 81 9A 09B9 1247 MOVAB PRC_T_OUTDVI(R11),R1 : GET DEVICE NAME DESCRIPTOR
28 A6 50 7D 09BE 1248 MOVZBL (R1)+,R0
09C1 1249 MOVA R0,SPWN_Q_OUTPUT(R6) : SAVE DEVICE NAME DESCRIPTOR
09C5 1250
09C5 1251
09C5 1252 : PROCESS LEGAL NON-RECORD-ORIENTED OUTPUT.
09C5 1253
05 40 A4 00 E0 09C5 1254 52$: BBS #DEV$V_REC,FAB$C_DEV(R4),56$ : ERROR IF NOT REC-ORIENTED
083B 30 09CA 1255 BSBW CREATE_OUTMBX : CREATE OUTPUT MAILBOX
1A 11 09CD 1256 BRB 91$ : DO NOT SET OUTPUT BIT
09CF 1257
09CF 1258 :

```

```

09CF 1259 : PROCESS RECORD-ORIENTED OUTPUT.
09CF 1260 :
52 28 A6 7D 09CF 1261 56$: MOVQ SPWN_Q_OUTPUT(R6),R2 : COPY DESCRIPTOR OF DEVICE NAME
54 68 7D 09D3 1262 : MOVQ (R8),R4 : COPY DESCRIPTOR OF SCRATCH AREA
07ED 30 09D6 1263 : BSBW GET_DEVICE : GET DEVICE NAME
28 A6 54 7D 09D9 1264 : MOVQ R4,SPWN_Q_OUTPUT(R6) : SAVE EQUIVALENCE STRING
68 54 A2 09DD 1265 : SUBW R4,(R8) : PERMANENTLY REMOVE RESULT
04 A8 54 C0 09E0 1266 : ADDL R4,4(R8) : FROM SCRATCH AREA
09E4 1267 :
09E4 1268 :
09E4 1269 : CHECK THAT OUTPUT FILE SPEC IS NOT TOO LONG FOR $CREPRC TO HANDLE.
09E4 1270 :
09E4 1271 90$: SETBIT SPWN_V_OUTPUT,SPWN_W_FLAGS(R6) : SET NO MAILBOX IN USE
5E 00B0 CE 9E 09E9 1272 91$: MOVAB FAB$C_BLN+NAM$C_BLN(SP),SP : DEALLOCATE FAB AND NAM
50 00038200 8F D0 09EE 1273 : MOVL #CLIS_INVFILSPEC,R0 : ASSUME SPEC IS TOO LONG
28 A6 00FF 8F B1 09F5 1274 : CMPW #LNMSC_NAMLENGTH,SPWN_Q_OUTPUT(R6) : IS IT REALLY?
03 1F 09FB 1275 : BLSSU 95$ : YES, THEN ERROR
09FD 1276 :
09FD 1277 :
09FD 1278 : EXIT WITH STATUS
09FD 1279 :
50 01 D0 09FD 1280 : MOVL #1,R0 : SET NORMAL STATUS
05 0A00 1281 95$: RSB : EXIT WITH STATUS

```

```

OA01 1283      .SBTTL ATTACH COMMAND
OA01 1284      :+
OA01 1285      DCL$ATTACH - ATTACH COMMAND
OA01 1286      :
OA01 1287      THIS ROUTINE IS CALLED TO EXECUTE THE DCL ATTACH COMMAND. THE ATTACH
OA01 1288      COMMAND TRANSFERS EXCLUSIVE CONTROL OF THE TERMINAL FROM THIS PROCESS
OA01 1289      TO ANOTHER PROCESS BY MUTUAL COOPERATION. A MESSAGE IS SENT TO THE
OA01 1290      'DESTINATION' PROCESS'S ATTACH REQUEST MAILBOX ASKING IF IT WANTS TO
OA01 1291      HAVE CONTROL OF THE TERMINAL. THE DESTINATION PROCESS RESPONDS WITH
OA01 1292      A YES/NO REPLY DEPENDING ON ITS STATE. IF IT RESPONDS WITH YES, THEN
OA01 1293      WE MARK OURSELVES DETACHED, AND WAIT FOR A RE-ATTACH REQUEST.
OA01 1294      :
OA01 1295      INPUTS:
OA01 1296      :
OA01 1297      R10 = ADDRESS OF COMMAND WORK AREA
OA01 1298      R11 = ADDRESS OF PROCESS WORK AREA
OA01 1299      :
OA01 1300      :
OA01 1301      OUTPUTS:
OA01 1302      :
OA01 1303      R0 = STATUS CODE
OA01 1304      :-
OA01 1305      :
OA01 1306      DCL$ATTACH::
OA01 1307      :
OA01 1308      GET THE PROCESS NAME OR PID FROM THE COMMAND LINE
OA01 1309      :
56      D4      OA01 1310      CLRL      R6      : MARK NO PROCESS NAME SUPPLIED YET
58      D4      OA03 1311      CLRL      R8      : MARK NO PID SUPPLIED YET
F5F8'   30      OA05 1312 40$:      BSBW      DCL$GETDVAL      : GET NEXT TOKEN
38 50    E9      OA08 1313      BLBC      R0,50$      : BRANCH IF END OF LINE
00 55    D1      OA0B 1314      CMPL      R5,#PTR_K_COMDQUAL      : VERB QUALIFIER?
03 55    13      OA0E 1315      BEQL      42$      : BRANCH IF SO
F0      D1      OA10 1316      CMPL      R5,#PTR_K_PARAMETR      : PARAMETER (PROCESS NAME)?
56 51    7D      OA15 1317      BNEQ      40$      : IF NOT, IGNORE IT
EB      11      OA18 1318      MOVQ      R1,R6      : SAVE DESCRIPTOR OF PROCESS NAME
F5E3'   30      OA1A 1319      BRB      40$
00'8F   91      OA1D 1320 42$:      BSBW      DCL$GETNVAL      : GET QUALIFIER NUMBER
E2      12      OA21 1321      CMPB      R1,#CLISK_ATTA_IDEN      : /IDENTIFICATION?
02 54    91      OA23 1322      BNEQ      40$      : IF NOT, IGNORE IT
DD      12      OA26 1323      CMPB      R4,#PTR_K_COLON      : IS A VALUE PRESENT?
F5D5'   30      OA28 1324      BNEQ      40$      : IF NOT, USE DEFAULT
52 51    7D      OA2B 1325      BSBW      DCL$GETDVAL      : GET /IDENT VALUE
51 00    9A      OA2E 1326      MOVQ      R1,R2      : PASS DESCRIPTOR OF VALUE STRING
F5CC'   30      OA31 1327      MOVZBL    #PRC_K_HEX,R1      : DEFAULT RADIX = HEX
05      12      OA34 1328      BSBW      DCL$CNVASCBIN      : CONVERT TO BINARY
58 51    D0      OA36 1329      BNEQ      45$      : BRANCH IF CONVERSION ERROR
CA      11      OA39 1330      MOVL      R1,R8      : SAVE PID AWAY
OA3B 1331      BRB      40$
OA42 1332 45$:      STATUS      EXPSYN      : EXPRESSION SYNTAX ERROR
05      1333      RSB
OA43 1334 50$:      BSBW      DCL$ATTACH2      : PERFORM ACTUAL ATTACH OPERATION
06 50    E9      OA45 1335      BLBC      R0,90$      : BRANCH IF ERROR DETECTED
03EB    30      OA48 1336      BSBW      RETURNED_MESSAGE      : WRITE MESSAGE SAYING WE HAVE RETURNED
50 01    D0      OA4B 1337      MOVL      #1,R0      : INDICATE SUCCESS
05      1338 90$:      RSB      : RETURN WITH STATUS

```

```

0A4F 1340 .SBTTL PERFORM ATTACH OPERATION
0A4F 1341 ---
0A4F 1342
0A4F 1343 ALTERNATE ENTRY POINT, USED BY ATTACH CLI CALLBACK. NO USE OF THE
0A4F 1344 WRK AREA MUST BE MADE AFTER THIS POINT, SINCE CLI CALLBACKS OPERATE
0A4F 1345 WITHOUT ANY SUCH AREA.
0A4F 1346
0A4F 1347 INPUTS:
0A4F 1348
0A4F 1349 R6/R7 = DESCRIPTOR OF PROCESS NAME
0A4F 1350 R8 = PID OF DESTINATION PROCESS
0A4F 1351 R11 = ADDRESS OF PRC AREA
0A4F 1352
0A4F 1353 EITHER THE PID OR PROCESS NAME MUST BE SPECIFIED.
0A4F 1354
0A4F 1355 OUTPUTS:
0A4F 1356
0A4F 1357 R0 = FINAL STATUS
0A4F 1358
0A4F 1359 ---
0A4F 1360
0A4F 1361 DCL$ATTACH2::
0A4F 1362
0A4F 1363 CONTROL/Y AST'S ARE DISABLED THROUGHOUT THIS COMMAND, TO ENSURE THAT
0A4F 1364 MAILBOXES WHICH ARE CREATED HERE ARE CORRECTLY DELETED, ETC. AS A
0A4F 1365 RESULT, WE MUST PERIODICALLY CHECK THE HANGUP FLAG IN CASE A HANGUP
0A4F 1366 IS DETECTED WHILE WE ARE OPERATING.
0A4F 1367
0A4F 1368 PUSHL PRC_L_OUTOFBAND(R11) ; SAVE OUT-OF-BAND ENABLE MASK
0A53 1369 CLRL R1 ; DISABLE ALL OUT-OF-BAND AST'S
0A55 1370 BSBW DCL$RESET00B
0A58 1371
0A58 1372
0A58 1373 LOOKUP THE DESTINATION PROCESS, AND IF PRESENT, GET THE PID AND MPID
0A58 1374
0A58 1375 CLRG -(SP) ; ALLOCATE AN IOSB
0A5A 1376 CLRG -(SP) ; CREATE GETJPI ITEM LIST
0A5C 1377 PUSHAB -2*4(SP) ; SET BUFFER ADDRESS
0A5F 1378 PUSHL #JPI$_PID@16+4 ; REQUEST PID, SET BUFFER LENGTH
0A65 1379 CLRL -(SP) ; SETUP MPID REQUEST ITEM
0A67 1380 PUSHAB -2*4(SP) ; SET BUFFER ADDRESS
0A6A 1381 PUSHL #JPI$_MASTER_PID@16+4 ; REQUEST MPID, SET BUFFER LENGTH
0A70 1382 PUSHR #*M<R6,R7,R8 ; PUSH PROCESS NAME DESCRIPTOR, PID
0A74 1383 MOVL SP,R0
0A77 1384 $GETJPIW S ITMLST=12(R0),- ; GET PID OF THE PROCESS
0A77 1385 IOSB=40(R0),-
0A77 1386 EFN=#EXESC $YSEFN,-
0A77 1387 PRCNAM=(R0),-
0A77 1388 PIDADR=8(R0)
0A91 1389 ADDL #3*4,SP ; POP PRCNAM DESCRIPTOR, PID
0A94 1390 POPL R3 ; GET THE DESTINATION MPID
0A97 1391 ADDL #2*4,SP ; POINT TO RETURNED DESTINATION PID
0A9A 1392 POPL R2 ; GET THE DESTINATION PID
0A9D 1393 ADDL #3*4,SP ; POP ITEM LIST
0AA0 1394 BLBC R0,49$ ; BRANCH IF ERROR
0AA3 1395 MOVZWL (SP),R0 ; GET IOSB STATUS
0AA6 1396 BLBS R0,55$ ; BRANCH IF PROCESS FOUND

```

```

5E 08 C0 0AA9 1397 49$: ADDL #8,SP ; POP THE IOSB
0220 31 0AAC 1398 BRW 95$ ; REPORT ERROR
0AAF 1399
0AAF 1400
0AAF 1401 : MAKE SURE WE AREN'T TRYING TO ATTACH TO OURSELVES
0AAF 1402
0AAF 1403 55$: CLRQ -(SP) ; CREATE GETJPI ITEM LIST
F8 AE 9F 0AB1 1404 PUSHAB -2*4(SP) ; SET BUFFER ADDRESS
03190004 8F DD 0AB4 1405 PUSHL #JPIS_PID@16+4 ; REQUEST OUR PID, SET BUFFER LENGTH
7E D4 0ABA 1406 CLRL -(SP) ; SETUP MPID REQUEST ITEM
F8 AE 9F 0ABC 1407 PUSHAB -2*4(SP) ; SET BUFFER ADDRESS
03250004 8F DD 0ABF 1408 PUSHL #JPIS_MASTER_PID@16+4 ; REQUEST OUR MPID, SET BUFFER LENGTH
50 5E D0 0AC5 1409 MOVL SP,R0 ; GET PID OF THIS PROCESS
0AC8 1410 $GETJPIW S ITMLST=(R0),-
0AC8 1411 IOSB=28(R0),-
0AC8 1412 EFN=#EXESC_SYSEFN ; GET OUR MPID
5E 54 8ED0 0AE0 1413 POPL R4 ; POINT TO OUR RETURNED PID
08 C0 0AE3 1414 ADDL #2*4,SP ; GET OUR PID
5E 51 8ED0 0AE6 1415 POPL R1 ; CLEANUP STACK
0C C0 0AE9 1416 ADDL #3*4,SP ; ASSUME AN ERROR
000388C2 8F D0 0AEC 1417 MOVL #CLIS_REFUSED,R0 ; TRYING TO ATTACH TO OURSELF?
52 51 D1 0AF3 1418 CMPL R1,R2 ; IF SO, REJECT THE OPERATION
B1 13 0AF6 1419 BEQL 49$ ; TRYING TO ATTACH TO A PROCESS OUTSIDE
54 53 D1 0AF8 1420 CMPL R3,R4 ; OF OUR JOB?
0AFB 1421 ; IF SO, REJECT THE OPERATION
5E AC 12 0AFB 1422 BNEQ 49$ ; POP THE IOSB
08 C0 0AFD 1423 ADDL #8,SP
0B00 1424
0B00 1425 : CREATE ATTACH$PID LOGICAL NAME FOR DESTINATION ATTACH MAILBOX
0B00 1426
0B00 1427
5E 1E C2 0B00 1428 SUBL #30,SP ; ALLOCATE ROOM FOR NAME
5E DD 0B03 1429 PUSHL SP ; CREATE DESCRIPTOR FOR NAME
1E DD 0B05 1430 PUSHL #30
51 F511 CF 9E 0B07 1431 MOVAB ATTACH_NAME,R1 ; GET ADDRESS OF ASCII FAO STRING
50 81 9A 0B0C 1432 MOVZBL (R1)+,R0 ; CONSTRUCT DESCRIPTOR OF STRING
7E 50 7D 0B0F 1433 MOVQ R0,-(SP) ; PUSH DESCRIPTOR ONTO STACK
50 5E D0 0B12 1434 MOVL SP,R0 ; CONSTRUCT LOGICAL NAME
0B15 1435 $FAO_S CTRSTR=(R0),-
0B15 1436 OUTBUF=8(R0),-
0B15 1437 OUTLEN=8(R0),-
0B15 1438 P1=R2 ; FROM PID FOR THIS PROCESS
5E 08 C0 0B26 1439 ADDL #8,SP ; POP FAO STRING DESCRIPTOR
0B29 1440
0B29 1441
0B29 1442 : GET SYSS$INPUT TRANSLATION (TERMINAL DEVICE NAME).
0B29 1443 : IF WE ARE AT CTRL/Y LEVEL, USE PROCESS PERMANENT SYSS$INPUT INSTEAD.
0B29 1444
54 10 3C 0B29 1445 60$: MOVZWL #ATTMBX_MAXMSG,R4 ; SET SIZE OF SCRATCH BUFFER
5E 54 C2 0B2C 1446 SUBL R4,SP ; ALLOCATE SCRATCH BUFFER FOR STRING
55 5E D0 0B2F 1447 MOVL SP,R5 ; CREATE DESCRIPTOR OF BUFFER
7E 52 7D 0B32 1448 MOVQ R2,-(SP) ; SAVE R2/R3
0B35 1449 BBC #PRC_V YLEVEL,- ; IF CTRL/Y LEVEL,
09 68 AB 0B37 1450 PRC_Q FLAGS(R11),62$ ; THEN SPECIAL CASE SYSS$OUTPUT
53 00000028 8F D0 0B3A 1451 MOVL #CTES$G_CLIDATA+PPDST_INPDVI,R3 ; GET ADDRESS OF ASCII DEVICE NAME
05 11 0B41 1452 BRB 64$ ; SKIP SYSS$INPUT PROCESSING
53 F4F3 CF 9E 0B43 1453 62$: MOVAB SYSS$INPUT,R3 ; CREATE SYSS$INPUT DESCRIPTOR

```

```

52 83 9A 0B48 1454 648: MOVZBL (R3)+,R2
0678 30 0B48 1455 BSBW GET_DEVICE
SC 50 E9 0B4E 1456 BLBC R0,6518
0B51 1457
0B51 1458
0B51 1459
0B51 1460
7E 7E 7C 0B51 1460 CLRQ -(SP)
7E 52 7D 0B53 1461 MOVQ R2,-(SP)
51 5E D0 0B56 1462 MOVL SP,R1
7E 7C 0B59 1463 CLRQ -(SP)
F8 AE 9F 0B5B 1464 PUSHAB -2*4(SP)
00040004 8F D0 0B5E 1465 PUSHL #DVIS_DEVCLASS@16+4
50 5E D0 0B64 1466 MOVL SP,R0
0B67 1467 $GETDVIW,S DEVNAM=(R1),-
0B67 1468 IOSB=8(R1),-
0B67 1469 EFN=#EXESC_SYSEFN,-
0B67 1470 ITMLST=(R0)
51 6E D0 0B82 1471 MOVL (SP),R1
5E 20 C0 0B85 1472 ADDL #4*8,SP
22 50 E9 0B88 1473 BLBC R0,6518
00000042 8F 51 D1 0B8D 1474 CLRL R0
02 13 0B94 1475 CMPL R1,#DCS_TERM
50 50 D6 0B96 1476 BEQL 6418
52 8E 7D 0B98 1477 INCL R0
0B98 1478 6418: MOVQ (SP)+,R2
0B98 1479
0B98 1480
0B98 1481
0B98 1482
7E 54 7D 0B98 1483 MOVQ R4,-(SP)
51 5E D0 0B9E 1484 MOVL SP,R1
13 50 E8 0BA1 1485 BLBS R0,668
0BA4 1486
FCDD 30 0BA4 1487 BSBW CHECK_TRMMBX
OD 50 E8 0BA7 1488 BLBS R0,668
0BA7 1489
011F 31 0BAA 1490 658: BRW 908
000388C2 8F D0 0BAD 1491 6518: MOVL #CLIS_REFUSED,R0
0115 31 0BB4 1492 BRW 908
0BB7 1493
0BB7 1494
0BB7 1495
0BB7 1496
0BB7 1497
0BB7 1498
0BB7 1499
0BB7 1500
0128 30 0BB7 1501 668: BSBW CREATE_ATTACHMBX
ED 50 E9 0BBA 1502 BLBC R0,658
0BB8 1503
0BB8 1504
0BB8 1505
0BB8 1506
50 7E DE 0BB8 1507 678: MOVAL -(SP),R0
0BC0 1508 $ASSIGN,S DEVNAM=4*8+ATTMBX_MAXMSG(R0),-
0BC0 1509 CHAN=(R0)
52 8ED0 0BCE 1510 POPL R2

```

: GET INPUT DEVICE NAME
: BRANCH IF ERROR

: CHECK IF WE ARE A TERMINAL

: ALLOCATE AN IOSB
: PUSH DEVICE DESCRIPTOR ON STACK
: GET ADDR. OF DESCRIPTOR
: CREATE AN ITEM LIST
: SET BUFFER ADDR.
: REQUEST DEVICE CLASS
: GET INPUT DEVICE CLASS

: GET DEVICE CLASS
: POP GETDVI ARGS. OFF THE STACK
: REFUSE ATTACH IF ERROR
: ASSUME IT IS A TERMINAL
: IS INPUT DEVICE A TERMINAL?
: BR IF YES. FLAG OK AS IS
: INDICATE NOT A TERMINAL
: RESTORE R2/R3

: CHECK FOR AN ASSOCIATED TERMINAL MAILBOX.

: PUSH SYSSINPUT TRANSLATION
: GET ADDRESS OF INPUT STRING
: DON'T CHECK FOR MAILBOX IF
: NOT A TERMINAL
: CHECK FOR MAILBOX
: BRANCH IF SUCCESS

: SET STATUS

: CREATE ATTACH MAILBOX FOR THIS PROCESS TO RECEIVE RE-ATTACH REQUESTS.
: IT IS DONE NOW, SO THAT ANY ERRORS DETECTED WILL CAUSE THE ENTIRE
: OPERATION TO FAIL, RATHER THAN SENDING THE ATTACH REQUEST TO THE
: OTHER PROCESS, AND THEN FINDING OUT WE CAN'T CREATE THE RE-ATTACH
: MAILBOX.

: CREATE ATTACH MAILBOX
: BRANCH IF ERROR DETECTED

: ASSIGN CHANNEL TO ATTACH MAILBOX

: ADDRESS OF LOGICAL NAME DESCRIPTOR
: ASSIGN CHANNEL TO MAILBOX
: GET CHANNEL OF MAILBOX

```

68 50 E9 OBD1 1511 BLBC R0,691$ ; BRANCH IF ERROR
OBD4 1512
OBD4 1513
OBD4 1514 : SEND REQUEST TO DESTINATION ATTACH REQUEST MAILBOX AND GET REPLY
OBD4 1515
OBD4 1516 68$: BSBW CHECK_FOR_HANGUP ; CHECK FOR HANGUP AST
54 05E6 30 OBD4 1516 MOVQA -(SP),R4 ; ALLOCATE IOSB FOR I/O REQUESTS
7E 7E OBD7 1517 $QIOW_S FUNC=#IOS_WRITEVBLK,- ; WRITE TO MAILBOX
OBD4 1518 CHAN=R2,-
OBD4 1519 EFN=#EXESC_SYSEFN,-
OBD4 1520 IOSB=(R4),-
OBD4 1521 P1=#12(R4),P2=8(R4)
OBD4 1522 : RECORD = SYSS$INPUT TRANSLATION
3B 50 E9 OBF8 1523 BLBC R0,69$ ; BRANCH IF ERROR
50 64 3C OBF8 1524 MOVZWL (R4),R0 ; GET FINAL STATUS FROM IOSB
35 50 E9 OC01 1525 BLBC R0,69$ ; BRANCH IF ERROR
53 7E 7E OC04 1526 MOVQA -(SP),R3 ; ALLOCATE RECORD BUFFER
04 A3 5B DO OC07 1527 MOVL R11,4(R3) ; GET PRC AREA ADDRS.
OC08 1528 $QIOW_S FUNC=#IOS_READVBLK,- ; READ RESPONSE FROM MAILBOX
OC08 1529 CHAN=R2,-
OC08 1530 EFN=#EXESC_SYSEFN,-
OC08 1531 IOSB=(R4),-
OC08 1532 ASTADR=READ_AST,-
OC08 1533 ASTPRM=R3,-
OC08 1534 P1=(R3),P2=#4
53 8ED0 OC30 1535 POPL R3 ; TO LONGWORD BUFFER
03 50 E9 OC33 1536 BLBC R0,69$ ; POP RECORD BUFFER
50 64 3C OC36 1537 MOVZWL (R4),R0 ; BRANCH IF ERROR
5E 0C CO OC39 1538 69$: ADDL #12,SP ; GET FINAL STATUS FROM IOSB
7C 50 E9 OC3C 1539 691$: BLBC R0,80$ ; POP THE IOSB
OC3F 1540 $DASSGN_S CHAN=R2 ; BRANCH IF ERROR
OC49 1541 ; DEASSIGN CHANNEL TO ATTACH MAILBOX
OC49 1542
OC49 1543 : IF DESTINATION PROCESS SAYS NO TO ATTACH REQUEST, REPORT IT TO THE USER
OC49 1544
79 53 E9 OC49 1545 BLBC R3,85$ ; BRANCH IF DESTINATION PROCESS SAYS NO
OC4C 1546
OC4C 1547
OC4C 1548
OC4C 1549 : WAIT FOR PROCESS TO RETURN CONTROL HERE (FROM ATTACH OR TERMINATION).
OC4C 1550 : IF WE ARE IN AN INTERACTIVE PROCESS, THEN TELL THE TERMINAL DRIVER THAT
OC4C 1551 : WE NOW OWN THE TERMINAL.
OC4C 1552
OC4C 1553
OC4C 1554 70$: BSBW CHECK_FOR_HANGUP ; CHECK FOR HANGUP AST
50 056E 30 OC4C 1554 MOVQA -(SP),R0 ; CREATE A TEMP. IOSB
7E 7E OC4F 1555 $QIOW_S FUNC=#IOS_SETMODE!IOSM_WRTATTN,- ; SET ATTN. AST ON MAILBOX
OC52 1556 CHAN=PRC_Q_ATTMBX(R11),-
OC52 1557 EFN=#EXESC_SYSEFN,-
OC52 1558 IOSB=(R0),-
OC52 1559 P1=ATTACH_AST,-
OC52 1560 P2=R11
5E 08 CO OC78 1561 ADDL #8,SP ; ADDRESS OF AST ROUTINE
OF EO OC82 1564 SHIBER_S ; PASS ADDRESS OF CLI STORAGE
C5 68 AB EO OC84 1565 BBS #PRC_V_DETACHED,- ; CLEAN UP STACK
06 EO OC87 1566 BBS #PRC_V_MODE,- ; HIBERNATE WAITING FOR WAKEUP
40 68 AB OC89 1567 PRC_Q_FLAGS(R11),90$ ; BRANCH IF STILL DETACHED
; SKIP IF NOT INTERACTIVE

```

```

50  7E  7C  OC8C  1568      CLRQ      -(SP)      ; ALLOCATE AN IOSB
    SE  DO  OC8E  1569      MOVL      SP,R0      ; GET ADDRESS OF IOSB
    OC91  1570      SQIOW,S  FUNC=#IOS,SETHODE!IOSM,TI,PROCESS,- ; ASSUME TERMINAL OWNERSHIP
    OC91  1571      OC91  1572      CHAN=PRC,D_INPCHAN(R11),-
    OC91  1573      IOSB=(R0),-
    SE  08  C0  OC83  1574      EFN=#EXESC,_SYSEFN
    50  01  DO  OC86  1575      ADDL      #8,SP      ; RESTORE THE STACK
    11  11  OC89  1576      MOVL      #1,R0      ; SET SUCCESSFUL
    OC8B  1577      BRB      90$      ; AND EXIT
    OC8B  1578      ;
    OC8B  1579      ; CLEANUP FROM ERROR DETECTED WHILE HAVING CHANNEL OPEN TO ATTACH MAILBOX
    OC8B  1580      ;
    50  000388C2 8F  DO  OC8B  1581 80$: $DASSGN,S CHAN=R2      ; DEASSIGN CHANNEL TO ATTACH MAILBOX
    OC8C  1582 85$: MOVL      #CLIS,_REFUSED,R0      ; ATTACH REQUEST REFUSED
    OC8C  1583
    OC8C  1584      ;
    OC8C  1585      ; RESTORE PROCESS CONTEXT
    OC8C  1586      ;
    SE  3E  C0  OC8C  1587 90$: ADDL      #8+ATTMBX_MAXMSG+8+30,SP; CLEAN STACK
    56  50  DO  OCCF  1588 95$: MOVL      R0,R6      ; SAVE FINAL STATUS CODE
    0144 30  OCD2  1589      BSBW      DEALLOC_SPWN      ; REMOVE INACTIVE SPWN BLOCKS
    008A 30  OCD3  1590      BSBW      DELETE_ATTMBX      ; DELETE OUR ATTACH MAILBOX
    51 8ED0  OCD8  1591      POPL      R1      ; RESTORE OUT-OF-BAND ENABLE MASK
    F322' 30  OCDB  1592      BSBW      DCL$RESET00B      ; RESTORE AST'S
    50  56  DO  OCDE  1593      MOVL      R6,R0      ; RESTORE EXIT STATUS
    05  05  OCE1  1594      RSB      ; EXIT

```

```

OCE2 1596 .SBTTL CREATE ATTACH REQUEST MAILBOX
OCE2 1597
OCE2 1598
OCE2 1599 THIS ROUTINE IS CALLED TO CREATE AN ATTACH REQUEST MAILBOX FOR
OCE2 1600 THIS PROCESS TO RECEIVE ATTACH REQUESTS FROM OTHER PROCESSES.
OCE2 1601
OCE2 1602 INPUTS:
OCE2 1603
OCE2 1604 R11 = ADDRESS OF PRC AREA
OCE2 1605
OCE2 1606 OUTPUTS:
OCE2 1607
OCE2 1608 R0 = STATUS CODE
OCE2 1609
OCE2 1610
OCE2 1611 CREATE_ATTMBX:
OCE2 1612 PUSH R2 ; SAVE REGISTERS
OCE2 1613 CLRQ -(SP) ; ALLOCATE AN IOSB
OCE2 1614
OCE2 1615 ; CREATE ATTACH MAILBOX AND LOGICAL NAME, SHOULD ANY OTHER PROCESS DESIRE
OCE2 1616 ; TO RE-ATTACH TO THIS PROCESS LATER ON.
OCE2 1617
OCE2 1618 CLRQ -(SP) ; CREATE GETJPI ITEM LIST
OCE2 1619 PUSHAB -2*4(SP) ; SET BUFFER ADDRESS
OCE2 1620 PUSHL #JPI$PID@16+4 ; REQUEST OUR PID, SET BUFFER LENGTH
OCE2 1621 MOVL SP,R0 ;
OCE2 1622 $GETJPIW S ITMLST=(R0),- ; GET PID OF THIS PROCESS
OCE2 1623 IOSB=16(R0),- ;
OCE2 1624 EFN=#EXESC_SYSEFN ;
OCE2 1625
OCE2 1626 POPL R2 ; GET OUR PID
OCE2 1627 ADDL #3*4,SP ; CLEANUP STACK
OCE2 1628 SUBL #30,SP ; ALLOCATE ROOM FOR NAME
OCE2 1629 PUSHL SP ; CREATE DESCRIPTOR FOR NAME
OCE2 1630 PUSHL #30 ;
OCE2 1631 MOVAB ATTACH_NAME,R1 ; GET ADDRESS OF ASCII FAO STRING
OCE2 1632 MOVZBL (R1)+,R0 ; CONSTRUCT DESCRIPTOR OF STRING
OCE2 1633 MOVQ R0,-(SP) ; PUSH DESCRIPTOR ONTO STACK
OCE2 1634 MOVL SP,R0 ;
OCE2 1635 $FAO_S CTRSTR=(R0),- ; CONSTRUCT LOGICAL NAME
OCE2 1636 OUTBUF=8(R0),- ;
OCE2 1637 OUTLEN=8(R0),- ;
OCE2 1638 P1=R2 ; FROM PID FOR THIS PROCESS
OCE2 1639 ADDL #8,SP ; POP FAO STRING DESCRIPTOR
OCE2 1640 MOVL SP,R0 ; ADDRESS OF LOGICAL NAME DESCRIPTOR
OCE2 1641 $CREMBX S (CHAN=PRC W ATTMBX(R11),- ; CREATE ATTACH MAILBOX
OCE2 1642 -BUFQUO=#ATTMBX_MAXMSG,- ; ONLY NEEDS TO HOLD 1 MESSAGE
OCE2 1643 PROMSK=#B1111T11100001111,- ; ONLY GIVE OWNER R/W ACCESS
OCE2 1644 LOGNAM=(R0) ;
OCE2 1645 ADDL #30+8,SP ; CLEANUP STACK
OCE2 1646 ADDL #8,SP ; POP THE IOSB
OCE2 1647 POPL R2 ; RESTORE REGISTERS
OCE2 1648 RSB

```

```

OD62 1649      .SBTTL  DELETE ATTACH REQUEST MAILBOX
OD62 1650      ---
OD62 1651      :
OD62 1652      : THIS ROUTINE IS CALLED TO DELETE THE ATTACH REQUEST MAILBOX
OD62 1653      : FOR THIS PROCESS.  THIS IS DONE AFTER WE COME BACK FROM DETACHED
OD62 1654      : STATE, SINCE AFTER THAT POINT, WE CAN NO LONGER ACCEPT ATTACH REQUESTS.
OD62 1655      :
OD62 1656      : INPUTS:
OD62 1657      :
OD62 1658      :     R11 = ADDRESS OF PRC AREA
OD62 1659      :
OD62 1660      : OUTPUTS:
OD62 1661      :
OD62 1662      :     NONE
OD62 1663      : ---
OD62 1664      :
OD62 1665      DELETE_ATTMBX:
OD62 1666      SDASSGN_S CHAN=PRC W ATTMBX(R11) ;DELETE ATTACH MAILBOX
7A AB  B4 OD6D 1667      CLRW  -PRC_W_ATTMBX(R11) ;MARK CHANNEL NO LONGER VALID
      05 OD70 1668      RSB

```

```

OD71 1670 .SBTTL CREATE TERMINATION MAILBOX
OD71 1671
OD71 1672
OD71 1673 THIS ROUTINE IS CALLED TO CREATE A TERMINATION MAILBOX FOR
OD71 1674 THIS PROCESS TO RECEIVE TERMINATION MESSAGE FROM UP TO FOUR OF ITS
OD71 1675 SPAWNED SUBPROCESSES.
OD71 1676
OD71 1677 INPUTS:
OD71 1678
OD71 1679 R6 = ADDRESS OF SPWN DATA STRUCTURE
OD71 1680 R8 = ADDRESS OF SCRATCH STORAGE
OD71 1681 R11 = ADDRESS OF PRC AREA
OD71 1682
OD71 1683 OUTPUTS:
OD71 1684
OD71 1685 R0 = STATUS CODE
OD71 1686 R7 = ADDRESS OF TMBX BLOCK
OD71 1687
OD71 1688
OD71 1689 CREATE_TMBX:
OD71 1690
OD71 1691
OD71 1692
OD71 1693
OD71 1694
OD71 1695
OD71 1696
OD71 1697
OD71 1698
OD71 1699
OD71 1700
OD71 1701
OD71 1702
OD71 1703
OD71 1704
OD71 1705
OD71 1706
OD71 1707
OD71 1708
OD71 1709
OD71 1710
OD71 1711
OD71 1712
OD71 1713
OD71 1714
OD71 1715
OD71 1716
OD71 1717
OD71 1718
OD71 1719
OD71 1720
OD71 1721
OD71 1722
OD71 1723
OD71 1724
OD71 1725
OD71 1726

51 10 3C OD71 1694 MOVZWL #TMBX_C_LENGTH,R1 ; LENGTH OF STORAGE TO ALLOCATE
F289 30 OD74 1695 BSBW DCL$ACLDYNMEM ; ALLOCATE STORAGE
78 50 E9 OD77 1696 BLBC R0,908 ; BRANCH ON ERROR
57 52 D0 OD7A 1697 MOVL R2,R7 ; SET ADDRESS OF TMBX BLOCK
OA A7 51 B0 OD7D 1698 MOVW R1,TMBX_W_SIZE(R7) ; STORE SIZE OF BLOCK
08 A7 94 OD81 1699 CLAB TMBX_B_REL'S(R7) ; SET REFERENCE COUNT TO ZERO
OC A7 5B D0 OD84 1700 MOVL R11,TMBX_L_PRC(R7) ; STORE BASE OF PROCESS WORK AREA
74 AB D0 OD88 1701 MOVL PRC_L_TMBX(R11),- ; INSERT BLOCK INTO LINKED LIST
67
74 AB 57 D0 OD8B 1702 TMBX_C_LINK(R7)
OD8C 1703 MOVL R7,PRC_L_TMBX(R11)
OD90 1704
OD90 1705
OD90 1706
OD90 1707
OD90 1708
OD90 1709
OD90 1710
OD90 1711
OD90 1712
OD90 1713
OD90 1714
OD90 1715
OD90 1716
OD90 1717
OD90 1718
OD90 1719
OD90 1720
OD90 1721
OD90 1722
OD90 1723
OD90 1724
OD90 1725
OD90 1726

4B 50 E9 ODAB 1711 BLBC R0,958 ; BRANCH ON ERROR
ODAB 1712 $QIOW,S FUNC=#IOS_SETMODE!IOSM_WRTATTN,- ; SET ATTENTION AST ON MAILBOX
ODAB 1713 CHAN=TMBX_W_CHANNEL(R7),-
ODAB 1714 IOSB=SPWN_Q_IOSB(R6),-
ODAB 1715 EFN=#EXESC_SYSEFN,-
ODAB 1716 P1=W^TERMINATION_AST,-
ODAB 1717 P2=R7 ; ADDRESS OF AST ROUTINE
ODAB 1718 BLBC R0,958 ; PASS TMBX BLOCK ADDRESS
ODAB 1719 MOVZWL SPWN_Q_IOSB(R6),R0 ; BRANCH IF ERROR
ODAB 1720 BLBC R0,958 ; GET IOSB STATUS
ODAB 1721
ODAB 1722
ODAB 1723
ODAB 1724
ODAB 1725
ODAB 1726

50 23 50 E9 ODD0 1718 BLBC R0,958 ; BRANCH IF ERROR
38 A6 3C ODD3 1719 MOVZWL SPWN_Q_IOSB(R6),R0 ; GET IOSB STATUS
1C 50 E9 ODD7 1720 BLBC R0,958 ; BRANCH IF ERROR
ODDA 1721
ODDA 1722
ODDA 1723
ODDA 1724
ODDA 1725
ODDA 1726

GET UNIT NUMBER OF TERMINATION MAILBOX
$GETCHN,S CHAN=TMBX_W_CHANNEL(R7),- ; GET MAILBOX INFORMATION
PRIBUF=(R8)

```

```

04 A7 06 50 E9 ODED 1727 BLBC R0,95$ : BRANCH ON ERROR
      14 A8 B0 ODF0 1728 MOVW DIB$W_UNIT+8(R8),TMBX_W_UNIT(R7) : SAVE MAILBOX UNIT NUMBER
      05 ODF5 1729 90$: RSB : RETURN WITH STATUS
      ODF6 1730 :
      ODF6 1731 :
      ODF6 1732 : DELETE TERMINATION MAILBOX AND TMBX DATA STRUCTURE AND THEN
      ODF6 1733 : RETURN WITH STATUS
      ODF6 1734 :
      50 DD ODF6 1735 95$: PUSHL R0 : SAVE STATUS
0004 30 ODF8 1736 BSBW DELETE_TMBX : DELETE ALLOCATED TMBX
      50 BED0 ODF8 1737 POPL R0 : RESTORE STATUS
      05 ODFF 1738 RSB : RETURN WITH STATUS

```

```

ODFF 1740      .SBTTL  DELETE TERMINATION MAILBOX
ODFF 1741      ---
ODFF 1742      :
ODFF 1743      : THIS ROUTINE IS CALLED TO DELETE A TERMINATION MAILBOX AND ITS ASSOCIATED
ODFF 1744      : TMBX DATA STRUCTURE.
ODFF 1745      :
ODFF 1746      : INPUTS:
ODFF 1747      :
ODFF 1748      :     R7 = ADDRESS OF TMBX DATA STRUCTURE
ODFF 1749      :     R11 = ADDRESS OF PRC AREA
ODFF 1750      :
ODFF 1751      : OUTPUTS:
ODFF 1752      :
ODFF 1753      :     NONE
ODFF 1754      : ---
ODFF 1755      :
ODFF 1756      : DELETE_TMBX:
ODFF 1757      : SDASSGN_S CHAN=TMBX_W CHANNEL(R7)      : DELETE TERMINATION MAILBOX
ODFF 1758      : MOVL      TMBX_L_LINK(R7),PRC_L_TMBX(R11) : REMOVE BLOCK FROM LIST
ODFF 1759      : MOVL      R7,R0      : SET ADDRESS OF BLOCK
ODFF 1760      : MOVZWL   TMBX_W_SIZE(R7),R1 : GET SIZE OF BLOCK
ODFF 1761      : BSBW      DCL$DEADYNMEM : DEALLOCATE BLOCK
ODFF 1762      :
ODFF 1763      :
74 AB 67 DO OE0A 1758
51 50 57 DO OE0E 1759
   OA A7 3C OE11 1760
   F1E8' 30 OE15 1761
        05 OE18 1762
        OE19 1763

```

```

OE19 1765 .SBTTL DEALLOCATE SPWN BLOCKS
OE19 1766 :---
OE19 1767 :
OE19 1768 : THIS ROUTINE IS CALLED TO DEALLOCATE THE SPWN BLOCKS OF TERMINATED
OE19 1769 : SUBPROCESSES.
OE19 1770 :
OE19 1771 : INPUTS:
OE19 1772 :
OE19 1773 : R11 = ADDRESS OF PRC AREA
OE19 1774 :
OE19 1775 : OUTPUTS:
OE19 1776 :
OE19 1777 : NONE
OE19 1778 :---
OE19 1779 :
50 00C0 CB DE OE19 1780 DEALLOC SPWN:
51 50 DO OE19 1781 MOVAL PRC L SPWN(R11),R0 : GET ADDR OF FIRST SPWN BLOCK
50 60 DO OE1E 1782 ASSUME SPWN L_LINK EQ 0 :
OF 13 OE1E 1783 10$: MOVL RO,RT : SAVE ADDR OF PREVIOUS SPWN BLOCK
07 E0 OE21 1784 MOVL SPWN_L_LINK(R0),R0 : GET NEXT SPWN BLOCK
F3 0C A0 OE24 1785 BEQL 90$ : BRANCH IF DONE
60 DO OE26 1786 BBS #SPWN V ACTIVE, - : BRANCH IF STILL IN USE
61 OE28 1787 SPWN_Q_FLAGS(R0),10$ :
51 04 A0 3C OE2B 1788 MOVL SPWN_L_LINK(R0), - : REMOVE BLOCK FROM LIST
F1CB' 30 OE2D 1789 SPWN_L_LINK(R1) :
05 OE2E 1790 MOVZWL SPWN_W_SIZE(R0),R1 : SET LENGTH OF BLOCK
OE32 1791 BSBW DCL$DEADYNMEM : DEALLOCATE BLOCK
OE35 1792 90$: RSB

```

```

OE36 1794 .SBTTL WRITE RETURNED MESSAGE
OE36 1795 ---
OE36 1796 :
OE36 1797 : THIS ROUTINE WRITES A MESSAGE INDICATING THAT TERMINAL CONTROL
OE36 1798 : HAS RETURNED TO THIS PROCESS.
OE36 1799 :
OE36 1800 : INPUTS:
OE36 1801 :
OE36 1802 :     NONE
OE36 1803 :
OE36 1804 : OUTPUTS:
OE36 1805 :
OE36 1806 :     NONE
OE36 1807 : ---
OE36 1808 :
OE36 1809 RETURNED_MESSAGE:
OE36 1810 :
OE36 1811 : OUTPUT MESSAGE SAYING THAT CONTROL OF THE TERMINAL HAS RETURNED
OE36 1812 : TO THIS PROCESS
OE36 1813 :
02 AE 5E 10 C2 OE36 1814      SUBL    #16,SP                :ALLOCATE SCRATCH BUFFER FOR PRCNAM
      5E 10 DD OE39 1815      PUSHL   SP                :CREATE DESCRIPTOR OF BUFFER
      50 5E DD OE3B 1816      PUSHL   #16
      7E 50 DD OE3D 1817      MOVL    SP,R0
      50 7E D4 OE40 1818      CLRL    -(SP)
      7E 60 7D OE42 1819      PUSHL   R0                :MARK END OF JPI LIST
      031C 8F B0 OE44 1820      MOVQ    (R0),-(SP)         :ADDRESS TO STORE RETURN LENGTH
      7E 7C 7C OE47 1821      MOVW    #JPI$_PRCNAM,2(SP)   :DESCRIPTOR OF STRING BUFFER
      50 5E D0 OE4D 1822      CLRQ    -(SP)               :SET PARAMETER ID TO RETRIEVE
      5E 5E D0 OE4F 1823      MOVL    SP,R0               :ALLOCATE AN IOSB
      5E 5E D0 OE52 1824      SGETJPIW S ITMLST=8(R0),-    :GET CURRENT PROCESS NAME
      5E 5E D0 OE52 1825      IOSB=(R0),-
      5E 5E D0 OE52 1826      EFN=#EXESC_SYSEFN
      5E 18 C0 OE6A 1827      ADDL    #6*4,SP             :POP ITEM LIST AND IOSB
      51 01 DD OE6D 1828      PUSHL   SP                :ADDRESS OF PROCESS NAME
      0003FD11 8F D0 OE6F 1829      MOVL    #1,R1         :SET NUMBER OF ARGS
      5E 18 C0 OE72 1830      MOVL    #CLIS_RETURNED,R0   :MESSAGE CODE
      5E 18 C0 OE79 1831      BSBW    DCL$FORMMSG         :OUTPUT MESSAGE
      5E 18 C0 OE7C 1832      ADDL    #8*16,SP            :CLEAN STACK
      05 05 OE7F 1833      RSB
      OE80 1834

```

```

      OE80 1836      .SBTTL WRITE CONTEXT TO SUBPROCESS
      OE80 1837      ---
      OE80 1838      :
      OE80 1839      : THIS ROUTINE WRITES THE PROCESS CONTEXT RECORDS TO THE SUBPROCESS
      OE80 1840      : INITIALIZATION ROUTINE VIA THE CONTEXT MAILBOX. EACH RECORD HAS
      OE80 1841      : A TYPE FIELD WHICH DISTINGUISHES THE DIFFERENT TYPES OF INFORMATION
      OE80 1842      : WHICH CAN BE PASSED FROM PARENT TO SUBPROCESS.
      OE80 1843      :
      OE80 1844      : INPUTS:
      OE80 1845      :
      OE80 1846      :     R6 = ADDRESS OF SPWN STORAGE
      OE80 1847      :
      OE80 1848      : OUTPUTS:
      OE80 1849      :
      OE80 1850      :     R0 = STATUS CODE
      OE80 1851      : ---
      OE80 1852      :
      OE80 1853      WRITE_CONTEXT:
      OE80 1854      PUSHF    #*M<R2,R3,R4,R5,R7>      ; SAVE REGISTERS
      OE84 1855      :
      OE84 1856      :
      OE84 1857      : CONSTRUCT THE HEADER CONTEXT RECORD, WHICH CONTAINS SUCH THINGS AS PROCESS
      OE84 1858      : PRIVILEGES, ETC.
      OE84 1859      :
      SE   FC00 CE   9E   OE84 1860      MOVAB    -CTX_C_MAXLEN(SP),SP      ; ALLOCATE BUFFER FOR CONTEXT RECORD
      57   5E   DO   OE89 1861      MOVL     SP,R7      ; SAVE ADDRESS OF BUFFER
      67   DO   3C   OE8C 1862      MOVZWL   #CTX_C_HEADER,CTX_W_TYPE(R7) ; SET TYPE OF RECORD
      7E   7C   OE8F 1863      CLRQ      -(SP)      ; SET END OF ITEM LIST, RETLEN ADDR
      02   A7   7F   OE91 1864      PUSHAQ   CTX_Q_PROCPRIV(R7)      ; SET ADDRESS OF BUFFER
      02040008 8F   DD   OE94 1865      PUSHL   #JPI$-PROCPRIV@16!8 ; SET GETJPI ITEM CODE AND LENGTH
      50   5E   DO   OE9A 1866      MOVL     SP,R0      ; ADDRESS OF ITEM LIST
      OE9D 1867      $GETJPIW S' ITMLST=(R0),-      ; GET INFORMATION
      OE9D 1868      IOSB=SPWN_Q_IOSB(R6),-
      OE9D 1869      EFN=#EXESC_SYSEFN
      SE   10   CO   OE85 1870      ADDL     #4*4,SP      ; POP GETJPI ITEM LIST
      OE   A7   94   OE88 1871      CLRB     CTX_B_FLAGS(R7)      ; CLEAR THE FLAGS BYTE
      OE8B 1872      :
      04   OC   A6   02   E1   OE8B 1873      BBC     #SPWN_V_WAIT,SPWN_W_FLAGS(R6),5$      ; BRANCH IF NOT SET
      OE8C 1874      SETBIT   CTX_V_WAIT,CTX_B_FLAGS(R7)      ;
      OE8C 1875      :
      04   OC   A6   03   E1   OE8C 1876      5$: BBC     #SPWN_V_AUTOLOGO,SPWN_W_FLAGS(R6),15$      ; BRANCH IF NOT SET
      OE8D 1877      SETBIT   CTX_V_AUTOLOGO,CTX_B_FLAGS(R7)      ;
      OE8D 1878      :
      04   OC   A6   04   E1   OE8D 1879      15$: BBC     #SPWN_V_MODE,SPWN_W_FLAGS(R6),16$      ; BRANCH IF NOT SET
      OE8E 1880      SETBIT   CTX_V_MODE,CTX_B_FLAGS(R7)      ;
      OE8E 1881      :
      04   68   AB   07   E1   OE8E 1882      16$: BBC     #PRC_V_VERIFY,PRC_W_FLAGS(R11),161$      ; COPY VERIFICATION FLAG
      OE8F 1883      SETBIT   CTX_V_VERIFY,CTX_B_FLAGS(R7)      ;
      OE8F 1884      :
      04   00AF   CB   07   E1   OE8F 1885      161$: BBC     #PRC_V_VERIMAGE,PRC_B_FLAGS2(R11),162$      ; COPY IMAGE VERIFICATION FL
      OE8F 1886      SETBIT   CTX_V_VERIMAGE,CTX_B_FLAGS(R7)      ;
      OE8F 1887      :
      12   OC   A6   09   E0   OE8F 1888      162$: BBS     #SPWN_V_PROMPT,SPWN_W_FLAGS(R6),17$      ; BRANCH IF EXPLICIT PROMPT
      50   00F0   CB   9A   OE8F 1889      MOVZBL   PRC_B_PROMPTLEN(R11),R0      ; GET PROMPT LENGTH
      OF   A7   50   90   OE8F 1890      MOVB     R0,CTX_B_PROMPTLEN(R7)      ;
      00F1   CB   50   28   OE8F 1891      MOVCL   R0,PRC_W_PROMPTCTRL(R11),-      ; GET PROMPT STRING
      10   A7   OE8F 1892      CTX_W_PROMPTCTRL(R7)      ;

```

```

50 00A2 C6 11 0EFE 1893 BRB 18$
OF A7 50 9A 0F00 1894 17$: MOVZBL SPWN_B_PROMPTLEN(R6),R0 ; GET PROMPT LENGTH
00A3 C6 50 90 0F05 1895 MOVB RO,CTX_B_PROMPTLEN(R7) ;
10 A7 28 0F09 1896 MOVCS RO,SPWN_Q_PMPCTRL(R6),- ; GET PROMPT STRING
OF0E 1897 CTX_W_PMPCTRL(R7) ;
OF10 1898
58 A6 D0 0F10 1899 18$: MOVL SPWN_L_OUTOFBAND(R6),- ; GET OUT-OF-BAND AST MASK
0A A7 OF13 1900 CTX_C_OUTOFBAND(R7) ;
OF15 1901
53 33 D0 0F15 1902 MOVL #CTX_C_HDRLEN,R3 ; COMPUTE SIZE OF RECORD
0200 30 0F18 1903 BSBW WRITE_MAILBOX ; WRITE THE RECORD TO THE MAILBOX
23 50 E9 0F1B 1904 BLBC RO,19$ ; BRANCH IF ERROR DETECTED
OF1E 1905
OF1E 1906 ;
OF1E 1907 ; WRITE THE COMMAND STRING RECORD TO THE MAILBOX
OF1E 1908 ;
30 A6 B5 0F1E 1909 TSTW SPWN_Q_CMDSTR(R6) ; ANY COMMAND STRING?
24 13 OF21 1910 BEQL 20$ ; BRANCH IF NOT
30 A6 B1 0F23 1911 CMPW SPWN_Q_CMDSTR(R6),- ; CHECK IF BIGGER THAN OUR BUFFER
03FE 8F OF26 1912 #CTX_C_MAXLEN-CTX_T_CMDSTR ;
06 18 OF29 1913 BLEQU 10$ ; BRANCH IF OK
03FE 8F 3C 0F2B 1914 MOVZWL #CTX_C_MAXLEN-CTX_T_CMDSTR,- ; ELSE, TRUNCATE STRING
30 A6 OF2F 1915 SPWN_Q_CMDSTR(R6) ;
67 01 B0 0F31 1916 10$: MOVW #CTX_C_CMDSTR,CTX_W_TYPE(R7) ; SET TYPE OF RECORD
30 A6 28 0F34 1917 MOVCS SPWN_Q_CMDSTR(R6),- ;
34 B6 OF37 1918 @SPWN_Q_CMDSTR+4(R6),- ;
02 A7 OF39 1919 CTX_T_CMDSTR(R7) ;
53 57 C2 0F3B 1920 SUBL R7,R3 ; COMPUTE LENGTH OF RECORD
01DA 30 0F3E 1921 BSBW WRITE_MAILBOX ; WRITE THE RECORD TO THE MAILBOX
03 50 E8 0F41 1922 19$: BLBS RO,20$ ; BRANCH IF SUCCESS
01CA 31 0F44 1923 BRW 90$
OF47 1924
OF47 1925 ;
OF47 1926 ; WRITE THE PROCESS LOGICAL NAMES TO THE MAILBOX
OF47 1927 ;
03 0C A6 06 E0 0F47 1928 20$: BBS #SPWN_V_LOGNAM,SPWN_W_FLAGS(R6),30$ ; BRANCH IF REQUESTED
0153 31 0F4C 1929 BRW 40$
OF4F 1930
OF4F 1931 ;
OF4F 1932 ; FIRST WRITE ALL OF THE TABLE NAMES
OF4F 1933 ;
67 05 B0 0F4F 1934 30$: MOVW #CTX_C_LNMTABLE,CTX_W_TYPE(R7) ; SET LOGICAL NAME TABLE
55 00000000 GF D0 0F52 1935 MOVL G^CTC$GL_LNMDIRECT,R5 ; GET PROCESS DIRECTORY
55 0C A5 D0 0F59 1936 MOVL LNMB$TABLE(R5),R5 ; GET IT'S TABLE
50 11 A5 D0 0F5D 1937 MOVL LNMT$C_CHILD(R5),R0 ; GET FIRST TABLE
55 50 D0 0F61 1938 31$: MOVL R0,R5 ; GET NEXT TABLE
54 09 A5 D0 0F64 1939 MOVL LNMT$C_NAME(R5),R4 ; GET POINTER TO LNMB
58 10 A4 01 E0 0F68 1940 BBS #LNMB$V-CONFINE,LNMB$B_FLAGS(R4),33$ ; SKIP SUBTREE IF CONFINED
01 0B A4 91 0F6D 1941 CMPB LNMB$B_ACMODE(R4),#PSL$C_EXEC ; PRIV MODE TABLE?
4C 1B 0F71 1942 BLEQU 32$ ; DON'T COPY IF SO.
04 A7 0B A4 90 0F73 1943 MOVB LNMB$B_ACMODE(R4),CTX_B_ACMODE(R7) ; COPY ACMODE
08 0B 88 0F78 1944 BICB3 #LNMB$B_TABLE,- ;
05 A7 10 A4 D0 0F7A 1945 LNMB$B_FLAGS(R4),CTX_B_TFLAGS(R7) ; COPY NAME FLAGS
08 A7 1D A5 D0 0F7E 1946 MOVL LNMT$C_BYTESLM(R5),CTX_L_QUOTA(R7) ; COPY QUOTA
53 0C A7 DE 0F83 1947 MOVAL CTX_T_LNMTABLE(R7),R3 ; PTR TO BUFFER
51 11 A4 DE 0F87 1948 MOVAL LNMB$T_NAME(R4),R1 ; PTR TO NAME
50 61 9A 0F8B 1949 MOVZBL (R1),R0 ; LENGTH OF NAME

```

```

50 D6 OF8E 1950 INCL R0 ; INCLUDE THE COUNT
30 BB OF90 1951 PUSHR #^M<R4,R5> ; SAVE REGS
63 61 50 28 OF92 1952 MOV C3 R0,(R1),(R3) ; COPY NAME
30 BA OF96 1953 POPR #^M<R4,R5> ; RESTORE REGS
51 0D A5 D0 OF98 1954 MOVL LNMTHSL_PARENT(R5),R1 ; PTR TO PARENT
51 09 A1 D0 OF9C 1955 MOVL LNMTHSL_NAME(R1),R1 ; PTR TO PARENT'S LNMB
51 11 A1 DE OFA0 1956 MOVAL LNMBSL_NAME(R1),R1 ; PTR TO NAME
50 61 9A OFA4 1957 MOVZBL (R1),R0 ; LENGTH OF TABLE NAME
30 D6 OFA7 1958 INCL R0 ; INCLUDE THE COUNT
63 61 50 BB OFA9 1959 PUSHR #^M<R4,R5> ; SAVE REGS
30 28 OFAB 1960 MOV C3 R0,(R1),(R3) ; COPY TABLE NAME
30 BA OFAF 1961 POPR #^M<R4,R5> ; RESTORE REGS
02 A7 53 57 C2 OFB1 1962 SUBL 2 R7,R3 ; COMPUTE SIZE OF CONTEXT RECORD
53 02 A3 OFB4 1963 SUBW 3 #CTX_W_ENTSIZE,R3,CTX_W_ENTSIZE(R7) ; SET SIZE OF ENTRY
01 5F 30 OFB9 1964 BSBW WRITE_MAILBOX ; WRITE THE RECORD TO THE MAILBOX
28 50 E9 OFBC 1965 BLBC R0,92$ ; QUIT ON ERROR
50 11 A5 D0 OFBF 1966 32$: MOVL LNMTHSL_CHILD(R5),R0 ; GET CHILD TABLE PTR
25 12 OFC3 1967 BNEQ 310$ ; AND PASS IT.
50 15 A5 D0 OFC5 1968 33$: MOVL LNMTHSL_SIBLING(R5),R0 ; SIBLING TABLE?
1F 12 OFC9 1969 BNEQ 310$ ; AND PASS IT.
55 0D A5 D0 OFCB 1970 MOVL LNMTHSL_PARENT(R5),R5 ; GET SIBLING OF PARENT TABLE
50 15 A5 D0 OFCF 1971 MOVL LNMTHSL_SIBLING(R5),R0 ; AND PASS IT
15 12 OFD3 1972 BNEQ 310$
OFD5 1973
OFD5 1974
OFD5 1975
OFD5 1976
54 67 06 D0 OFD5 1977 MOVL #CTX_C_LNMMNAME,CTX_W_TYPE(R7) ; SET LOGICAL NAME TYPE
00000000'GF D0 OFD8 1978 MOVL G^CT$GL_LNMMHASH,R7 ; GET POINTER TO HASH TABLE
54 64 D2 OFDF 1979 MCOML LNMHSHSL_MASK(R4),R4 ; GET SIZE MASK
54 D6 OFE2 1980 INCL R4 ; CONVERT TO SIZE
00AC 31 OFE4 1981 34$: BRW 38$ ; LOOP
OFE7 1982
0127 31 OFE7 1983 92$: BRW 90$ ;HELPER BRANCH
FF74 31 OFEA 1984 310$: BRW 31$ ;HELPER BRANCH
OFED 1985
55 65 D0 OFED 1986 35$: MOVL (R5),R5 ; GET NEXT LINK IN CHAIN
F2 13 OFF0 1987 BEQL 34$ ; GO TO NEXT CHAIN IF NONE
F6 10 A5 01 E0 OFF2 1988 BBS #LNMB$V_CONFINE,LNMB$B_FLAGS(R5),35$ ; SKIP NAME IF CONFINED
F1 10 A5 03 E0 OFF7 1989 BBS #LNMB$V_TABLE,LNMB$B_FLAGS(R5),35$ ; SKIP NAME IF TABLE
01 0B A5 91 OFFC 1990 CMPB LNMB$B_ACMODE(R5),#PSL$C_EXEC ; PRIV MODE TABLE?
04 A7 0B A5 90 1000 1991 BLEQU 35$ ; DON'T COPY IF SO.
05 A7 10 A5 90 1007 1992 MOV B LNMB$B_ACMODE(R5),CTX_B_ACMODE(R7) ; COPY ACMODE
53 07 A7 DE 100C 1994 MOVB LNMB$B_FLAGS(R5),CTX_B_RFLAGS(R7) ; COPY NAME FLAGS
51 0C A5 D0 1010 1995 MOVAL CTX_T_LNMMNAME(R7),R3 ; PTR TO BUFFER
51 09 A1 D0 1014 1996 MOVL LNMB$C_TABLE(R5),R1 ; PTR TO TABLE
50 11 A1 9A 1018 1997 MOVZBL LNMTHSL_NAME(R1),R1 ; PTR TO TABLE'S LNMB
30 D6 101C 1998 INCL R0 ; LENGTH OF TABLE NAME
63 11 A1 50 28 1020 2000 PUSHR #^M<R4,R5> ; INCLUDE THE COUNT
30 BA 1025 2001 MOV C3 R0,LNMBST_NAME(R1),(R3) ; SAVE REGS
50 11 A5 9E 1027 2002 POPR #^M<R4,R5> ; COPY NAME
52 D4 102B 2003 MOVAB LNMBST_NAME(R5),R0 ; RESTORE REGS
0A 11 102D 2004 CLRL R2 ; PTR TO NAME
0E 60 02 E0 102F 2005 350$: BRS 351$ ; COUNT # OF XLATIONS
52 D6 1033 2006 INCL R2 ; TRY NEXT ENTRY
; BR IF END OF LIST
; COUNT THIS XLATION

```

```

50 04 A0 9E 1035 2007      MOVAB  LNMHST_XLATION(R0),R0      : POINT TO STRING
51 80 9A 1039 2008 351$: MOVZBL  (R0)+,R1      : GET LENGTH OF THIS STRING
50 51 C0 103C 2009      ADDL2  R1,R0      : SKIP OVER IT
51 52 11 103F 2010      BRB  350$      : TRY NEXT ENTRY
06 A7 52 90 1041 2011 352$: MOVB  R2,CTX_B_TRUNCNT(R7)      : SAVE COUNT
51 11 A5 9E 1045 2012      MOVAB  LNMHST_NAME(R5),R1      : ADDR OF NAME AGAIN
50 50 D6 1049 2013      INCL  R0      : COUNT TERMINATION BYTE
52 0400 C7 9E 104E 2015      MOVAB  CTX_C_MAXLEN(R7),R2      : COMPUTE SIZE NEEDED
52 53 C2 1053 2016      SUBL2  R3,R2      : GET END ADDR OF CTX
52 50 D1 1056 2017      CMPL  R0,R2      : BUFFER SPACE REMAINING
51 18 1A 1059 2018      BGTRU  36$      : NAME ALL FIT?
63 61 30 BB 105B 2019      PUSHR  #A<R4,R5>      : NOPE
53 30 28 105D 2020      MOVCS  R0,(R1),(R3)      : SAVE REGS
53 30 BA 1061 2021      POPR  #A<R4,R5>      : COPY NAME AND XLATIONS
02 A7 53 57 C2 1063 2022      SUBL2  R7,R3      : RESTORE REGS
53 02 A3 1066 2023      SUBW3  #CTX_W_ENTSIZE,R3,CTX_W_ENTSIZE(R7) : COMPUTE SIZE OF CONTEXT RECORD
50 00AD 30 106B 2024      BSBW  WRITE_MAILBOX      : SET SIZE OF ENTRY
75 50 E9 106E 2025      BLBC  R0,49$      : WRITE THE RECORD TO THE MAILBOX
51 1D 11 1071 2026      BRB  37$      : QUIT ON ERROR
1073 2027      : ONTO THE NEXT NAME
1073 2028      :
1073 2029      : DEAL WITH RECORDS TOO LONG FOR ONE MAILBOX MESSAGE
1073 2030      : (FOR NOW, ISSUE THE "SYMBOL TOO LONG" ERROR.
1073 2031      :
18 0C A6 00 E1 1073 2032 36$: BBC  #SPWN_V_LOG,SPWN_W_FLAGS(R6),37$ : BRANCH IF /NOLOG SPECIFIED
51 01 A1 9F 1078 2033      PUSHAB 1(R1)      : PUSH DESCRIPTOR OF THE
7E 61 9A 107B 2034      MOVZBL (R1),-(SP)      : LOGICAL NAME
51 5E DD 107E 2035      PUSHL  SP      : ADDRESS OF THE DESCRIPTOR
50 00038218 BF DO 1080 2036      MOVL  #1,R1      : SET NUMBER OF ARGS
51 EF73 30 DO 1083 2037      MOVL  #CLIS_SYMTOOLNG,R0      : MESSAGE CODE
5E 08 C0 108A 2038      BSBW  DCL$FORMMSG      : OUTPUT MESSAGE
FF5A 31 C0 108D 2039      ADDL  #4*2,SP      : CLEAN STACK
55 00000000 GF DO 1090 2040 37$: BRW  35$      : PROCESS NEXT SYMBOL
55 08 A544 DE 1093 2041      :
55 EE 54 F4 1093 2042 38$: MOVL  G^CTL$GL_LNMHASH,R5      : GET ADDR OF HASHTABLE
10A2 2043      MOVAL  LNMHSH$C_BUCKET-4(R5)[R4],R5      : GET ADDR OF BUCKET
10A2 2044      SOBGEQ  R4,37$      : LOOP OVER ALL CHAINS
10A2 2045      :
10A2 2046      : WRITE THE DCL GLOBAL SYMBOLS TO THE MAILBOX
10A2 2047      :
10A2 2048      :
16 0C A6 D5 E1 10A2 2049 40$: BBC  #SPWN_V_CLISYM,SPWN_W_FLAGS(R6),45$ : BRANCH IF NOT REQUESTED
54 28 AB 9E 10A7 2050      MOVAB  PRC_Q_GLOBAL(R11),R2      : GET ADDRESS OF GLOBAL LISTHEAD
04 A7 00 90 10AB 2051      MOVB  #CTX_C_GLOBAL,CTX_B_SYMTAB(R7) : SET WHICH SYMBOL TABLE
54 0097 30 10AF 2052      BSBW  WRITE_SYMBOLS      : WRITE ALL GLOBAL SYMBOLS
04 38 AB 9E 10B2 2053      MOVAB  PRC_Q_LOCAL(R11),R4      : GET ADDRESS OF LOCAL LISTHEAD
04 A7 01 90 10B6 2054      MOVB  #CTX_C_LOCAL,CTX_B_SYMTAB(R7) : SET WHICH SYMBOL TABLE
008C 30 10BA 2055      BSBW  WRITE_SYMBOLS      : WRITE ALL LOCAL SYMBOLS
10BD 2056      :
10BD 2057      : WRITE THE KEYPAD STATE TO THE MAILBOX
10BD 2058      :
10BD 2059      :
27 0C A6 0C E1 10BD 2060 45$: BBC  #SPWN_V_KEYPAD,SPWN_W_FLAGS(R6),50$ : BRANCH IF NOT REQUESTED
54 40 AB 9E 10C2 2061      MOVAB  PRC_Q_KEYPAD(R11),R2      : GET ADDRESS OF KEYPAD LISTHEAD
04 A7 02 90 10C6 2062      MOVB  #CTX_C_KEYTAB,CTX_B_SYMTAB(R7) : SET WHICH SYMBOL TABLE
007C 30 10CA 2063      BSBW  WRITE_SYMBOLS      : WRITE ALL KEYPAD SYMBOLS

```

```

      67 04 B0 10CD 2064      MOVW  #CTX_C_KEYSTATE,CTX_W_TYPE(R7)  : SET TYPE OF RECORD
52  4C AB D0 10D0 2065      MOVL  PRC_C_LASTKEY(R1),R2-      : GET ADDR OF ASCII KEY STATE
      51 82 9A 10D4 2066      MOVZBL (R2)+,R1      : GET DESCRIPTOR
03 02 A7 51 90 10D7 2067      MOVW  R1,CTX_B_KEYLENGTH(R7)      : COPY THE LENGTH
      62 51 28 10DB 2068      MOVC  R1,(R2),CTX_T_KEYSTATE(R7)  : COPY THE STRING
      53 57 C2 10E0 2069      SUBL  R7,R3      : COMPUTE LENGTH OF RECORD
      0035 30 10E3 2070      BSBW  WRITE_MAILBOX      : WRITE THE RECORD TO THE MAILBOX
      28 50 E9 10E6 2071 498: BLBC  R0,908      : BRANCH IF ERROR DETECTED
      10E9 2072      :
      10E9 2073      :
      10E9 2074      : WRITE AN END-OF-FILE TO THE MAILBOX AND WAIT FOR COMPLETION, SO THAT
      10E9 2075      : WE DON'T DELETE THE MAILBOX BEFORE THE SUBPROCESS EVEN GETS A CHANCE
      10E9 2076      : TO ASSIGN A CHANNEL TO IT.
      10E9 2077      :
      10E9 2078 508:  $QIOW_S FUNC=#IOS_WRITEOF,-      : WRITE AN EOF TO MAILBOX
      10E9 2079      : CHAN=SPWN_W_CHAN(R6),-
      10E9 2080      : IOSB=SPWN_Q_IOSB(R6),-
      10E9 2081      : EFN=#EXESC_SYSEFN
      10E9 2082      :
      50  04 50 E9 110A 2082      BLBC  R0,908      : BRANCH IF ERROR
      5E  38 A6 3C 110D 2083      MOVZWL SPWN_Q_IOSB(R6),R0      : GET IOSB STATUS
      0400 CE 9E 1111 2084 908:  MOVAB CTX_C_MAXLEN(SP),SP      : DEALLOCATE RECORD BUFFER
      00BC 8F BA 1116 2085      POPR  #MZR2,R3,R4,R5,R7>      : RESTORE REGISTERS
      05 111A 2086      RSB

```

```

111B 2088 .SBTTL WRITE RECORD TO CONTEXT MAILBOX
111B 2089 ---
111B 2090
111B 2091 WRITE A RECORD TO THE MAILBOX WITHOUT WAITING FOR A READER
111B 2092
111B 2093 INPUTS:
111B 2094
111B 2095 R6 = ADDRESS OF SPWN STORAGE
111B 2096 R7 = ADDRESS OF RECORD TO BE OUTPUT
111B 2097 R3 = LENGTH OF RECORD TO BE OUTPUT
111B 2098
111B 2099 OUTPUTS:
111B 2100
111B 2101 R0 = FINAL STATUS
111B 2102 ---
111B 2103
111B 2104 WRITE_MAILBOX:
009F 30 111B 2105 BSBW CHECK FOR HANGUP ; CHECK FOR HANGUP AST
111E 2106 $Q10W_S FUNC=7IOS_WRITEVBLK!IOSM_NOW,- ; WRITE TO MAILBOX WITHOUT WAITING
111E 2107 CHAN=SPWN_W CHAN(R6),-
111E 2108 EFN=#EXESC SYSEFN,-
111E 2109 IOSB=SPWN_Q IOSB(R6),-
111E 2110 P1=(R7),P2=R3 ; ADDRESS/LENGTH OF BUFFER
50 04 50 E9 1141 2111 R0,90$ ; BRANCH IF ERROR DETECTED
38 A6 3C 1144 2112 SPWN_Q_IOSB(R6),R0 ; GET FINAL STATUS
05 1148 2113 90$: RSB

```

```

1149 2115 .SBTTL WRITE ALL SYMBOLS IN A SYMBOL TABLE
1149 2116 ---
1149 2117
1149 2118 WRITE ALL THE SYMBOL RECORDS TO THE MAILBOX FROM A
1149 2119 SPECIFIED SYMBOL TABLE.
1149 2120
1149 2121 INPUTS:
1149 2122
1149 2123 CTX_B SYMTAB(R7) = TYPE OF SYMBOL TABLE
1149 2124 R4 = ADDRESS OF SYMBOL TABLE LISTHEAD
1149 2125 R6 = ADDRESS OF SPWN STORAGE
1149 2126 R7 = ADDRESS OF CONTEXT RECORD BUFFER
1149 2127
1149 2128 OUTPUTS:
1149 2129
1149 2130 RO-R5 DESTROYED.
1149 2131 ---
1149 2132
1149 2133 WRITE_SYMBOLS:
1149 2134 MOVW #CTX_C_CLISYM,CTX_W_TYPE(R7) ; SET TYPE OF RECORD
1149 2135 MOVW R4,R5 ; SAVE FOR END-OF-LIST CHECK
1149 2136 45$: MOVL (R4),R4 ; GET NEXT SYMBOL ENTRY IN LIST
1149 2137 CMPL R4,R5 ; END OF LIST?
1149 2138 BEQL 50$ ; BRANCH IF SO
1149 2139 ASSUME SYM_K_STRING EQ CTX_C_STRING
1149 2140 ASSUME SYM_K_PERM EQ CTX_C_PERM
1149 2141 ASSUME SYM_K_BINARY EQ CTX_C_BINARY
1149 2142 ASSUME SYM_K_KEYPAD EQ CTX_C_KEYPAD
1149 2143 MOVW SYM_B_TYPE(R4),CTX_B_SYMTYPE(R7) ; COPY SYMBOL TYPE CODE
1149 2144 MOVW SYM_B_NONUNIQUE(R4),CTX_B_NONUNIQUE(R7) ; COPY UNIQUENESS POINT
1149 2145 MOVZBL SYM_T_SYMBOL(R4),R0 ; GET LENGTH OF SYMBOL NAME
1149 2146 CMPB SYM_B_TYPE(R4),#SYM_K_BINARY ; BINARY VALUE?
1149 2147 BNEQ 47$ ; BRANCH IF NOT
1149 2148 ADDL #4+1,R0 ; COMPUTE LENGTH OF ASCII NAME + VALUE
1149 2149 BRB 48$
1149 2150 47$: MOVAB SYM_T_SYMBOL+1(R4)(R0),R1 ; ADDRESS OF WORD-COUNTED VALUE STRING
1149 2151 MOVZWL (R1),R1 ; GET LENGTH OF VALUE STRING
1149 2152 MOVAB 3(R1)(R0),R0 ; COMPUTE LENGTH OF ASCII NAME + VALUE
1149 2153 48$: CMPW R0,#<CTX_C_MAXLEN-CTX_T_SYMBOL> ; WILL SYMBOL VALUE FIT?
1149 2154 BGTRU 60$ ; IF LARGER, THEN NO
1149 2155 PUSHR #^M<R4,R5> ; SAVE REGISTERS
1149 2156 MOVC R0,SYM_T_SYMBOL(R4),CTX_T_SYMBOL(R7) ; MOVE NAME + VALUE
1149 2157 POPR #^M<R4,R5> ; RESTORE REGISTERS
1149 2158 SUBL R7,R3 ; COMPUTE SIZE OF RECORD
1149 2159 SUBW3 #CTX_W_ENTSIZE,R3,CTX_W_ENTSIZE(R7) ; SET SIZE OF ENTRY
1149 2160 BSBB WRITE_MAILBOX ; WRITE RECORD TO MAILBOX
1149 2161 BRB 45$ ; LOOP UNTIL TABLE EXHAUSTED
1149 2162 50$: RSB
1149 2163
1149 2164 :
1149 2165 : WE HAVE ENCOUNTERED AN OVERSIZED SYMBOL. ISSUE A WARNING MESSAGE SAYING
1149 2166 : THAT WE ARE IGNORING IT AND THEN PROCEED.
1149 2167 :
1149 2168 60$: BBC #SPWN_V_LOG,SPWN_W_FLAGS(R6),45$ ; BRANCH IF /NOLOG SPECIFIED
1149 2169 MOVAB SYM_T_SYMBOL+1(R4),-(SP) ; PUSH DESCRIPTOR OF THE
1149 2170 MOVZBL SYM_T_SYMBOL(R4),-(SP) ; SYMBOL NAME
1149 2171 PUSHL SP ; ADDRESS OF THE DESCRIPTOR

```

50	51	01	D0	11AA	2172	MOVL	#1,R1	:	SET NUMBER OF ARGS
	00038218	8F	D0	11AD	2173	MOVL	#CLIS SYMTOOLNG,R0	:	MESSAGE CODE
		EE49	30	11B4	2174	BSBW	DCL\$FORMMSG	:	OUTPUT MESSAGE
	SE	08	CO	11B7	2175	ADDL	#4*2,SP	:	CLEAN STACK
		FF92	31	11BA	2176	BRW	458	:	PROCESS NEXT SYMBOL
				11BD	2177				

```

1180 2179 .SBTTL CHECK FOR PENDING HANGUP AST
1180 2180 ---
1180 2181
1180 2182 THIS ROUTINE IS CALLED TO CHECK IF A HANGUP AST HAS COME IN SINCE
1180 2183 WE STARTED THE COMMAND EXECUTION. IF SO, THE PROCESS IS TERMINATED.
1180 2184
1180 2185 INPUTS:
1180 2186
1180 2187 R11 = ADDRESS OF PRC AREA
1180 2188
1180 2189 OUTPUTS:
1180 2190
1180 2191 NONE
1180 2192 ---
1180 2193
1180 2194 CHECK_FOR_HANGUP:
1180 2195 BBC #PRC_V_HANGUP,PRC_W_FLAGS(R11) 90$ : BRANCH IF NO HANGUP PENDING
1180 2196 BRW DCL$RESTART ; ABORT PROCESS
1180 2197 90$: RSB
03 68 AB OC E1
EE3B' 31
05 11C2 2196
11C5 2197 90$:

```

```

11C6 2199      .SBTTL GET DEVICE NAME
11C6 2200      ----
11C6 2201
11C6 2202      THIS ROUTINE CALLS THE $GETDVI SYSTEM SERVICE TO GET THE CANNONICAL NAME OF
11C6 2203      A SPECIFIED DEVICE. IT PLACES THE DEVICE NAME IN THE SPECIFIED BUFFER AND
11C6 2204      PROPAGATES THE STATUS RETURNED BY $GETDVI TO THE CALLER.
11C6 2205
11C6 2206      INPUTS:
11C6 2207
11C6 2208      R2/R3 = DESCRIPTOR OF DEVICE NAME
11C6 2209      R4/R5 = DESCRIPTOR OF RETURN BUFFER
11C6 2210
11C6 2211      OUTPUTS:
11C6 2212
11C6 2213      R0 = STATUS RETURNED BY $GETDVI
11C6 2214
11C6 2215      ----
11C6 2216
11C6 2217      GET_DEVICE:
11C6 2218      CLRQ      -(SP)
11C6 2219      MOVQ      R2,-(SP)
11CB 2220      MOVL      SP,R1
11CE 2221      CLRL      -(SP)
11D0 2222      PUSHAB   -12(SP)
11D3 2223      MOVQ      R4,-(SP)
11D6 2224      MOVW      #DVI$_DEVNAM,2(SP)
11DA 2225      MOVL      SP,R0
11DB 2226      $GETDVIW S' DEVNAM=(R1),-
11DD 2227      IOSB=8(R1),-
11DD 2228      EFN=#EXESC,SYSEFN,-
11DD 2229      ITMLST=(R0)
11F8 2230      MOVZWL   (SP),R4
11FB 2231      ADDL      #6*4,SP
11FE 2232      BLBC      R0,90$
1201 2233      MOVZWL   (SP),R0
1204 2234      ADDL      #8,SP
1207 2235      RSB
90$:
54 7E 7C 11C6 2218      CLRQ      -(SP)
51 5E 7D 11C8 2219      MOVQ      R2,-(SP)
F4 7E D4 11CB 2220      MOVL      SP,R1
AE 54 9F 11CE 2221      CLRL      -(SP)
02 7E 7D 11D0 2222      PUSHAB   -12(SP)
AE 54 7D 11D3 2223      MOVQ      R4,-(SP)
50 AE 20 B0 11D6 2224      MOVW      #DVI$_DEVNAM,2(SP)
50 5E D0 11DA 2225      MOVL      SP,R0
11DB 2226      $GETDVIW S' DEVNAM=(R1),-
11DD 2227      IOSB=8(R1),-
11DD 2228      EFN=#EXESC,SYSEFN,-
11DD 2229      ITMLST=(R0)
54 6E 3C 11F8 2230      MOVZWL   (SP),R4
5E 18 C0 11FB 2231      ADDL      #6*4,SP
03 50 E9 11FE 2232      BLBC      R0,90$
50 6E 3C 1201 2233      MOVZWL   (SP),R0
5E 08 C0 1204 2234      ADDL      #8,SP
05 05 1207 2235      RSB

```

```

: ALLOCATE AN IOSB
: PUSH DEVICE DESCRIPTOR ON STACK
: GET ADDRESS OF DESCRIPTOR
: CREATE GETDVI ITEM LIST
: RETURN LENGTH ON STACK
: PUSH BUFFER DESCRIPTOR ON STACK
: REQUEST COMPLETE DEVICE NAME
: GET ADDRESS OF ITEM LIST
: GET TERMINAL DEVICE NAME
:
: GET LENGTH OF RESULT
: RESTORE STACK
: BRANCH IF ERROR
: GET IOSB STATUS
: POP THE IOSB

```

```

1208 2237 .SBTTL CREATE OUTPUT MAILBOX
1208 2238 ---
1208 2239
1208 2240 THIS ROUTINE IS CALLED TO CREATE AN OUTPUT MAILBOX FOR THIS PROCESS TO
1208 2241 RECEIVE WRITE REQUESTS FROM ITS SPAWNED SUBPROCESSES.
1208 2242
1208 2243 INPUTS:
1208 2244
1208 2245 R6 = ADDRESS OF SPWN DATA STRUCTURE
1208 2246 R8 = ADDRESS OF SCRATCH AREA
1208 2247 R11 = ADDRESS OF PRC DATA STRUCTURE
1208 2248
1208 2249 OUTPUTS:
1208 2250
1208 2251 RO = STATUS CODE
1208 2252 ---
1208 2253
1208 2254 CREATE_OUTMBX:
1208 2255
1208 2256
1208 2257 GET PID OF PARENT.
1208 2258
1208 2259 CLRQ -(SP)
1208 2260 PUSHAB -8(SP)
1208 2261 PUSHL #JPI$_PID@16+4
1208 2262 MOVL SP,RO
1208 2263 SGETJPIW S ITMLST=(RO),-
1208 2264 EFH=#EXESC SYSEFN,-
1208 2265 IOSB=SPWN_Q_IOSB(R6)
1208 2266 POPL RO
1208 2267 ADDL #3*4,SP
1208 2268
1208 2269
1208 2270 CREATE MAILBOX LOGICAL NAME.
1208 2271
1208 2272 MOVQ (R8),-(SP)
1208 2273 MOVAB OUTPUT_NAME,R2
1208 2274 MOVZBL (R2)+,R1
1208 2275 MOVQ R1,-(SP)
1208 2276 MOVL SP,R1
1208 2277 $FAO_S CTRSTR=(R1),-
1208 2278 OUTBUF=8(R1),-
1208 2279 OUTLEN=8(R1),-
1208 2280 P1=RO
1208 2281 ADDL #8,SP
1208 2282 MOVQ (SP)+,SPWN_Q_OUTPUT(R6)
1208 2283 SUBL SPWN_Q_OUTPUT(R6),(R8)
1208 2284 ADDL SPWN_Q_OUTPUT(R6),4(R8)
1208 2285 BLBS RO,20$
1208 2286 BRW 91$
1208 2287 10$: BRW 90$
1208 2288
1208 2289
1208 2290 CHECK FOR ALREADY EXISTING MAILBOX.
1208 2291
1208 2292 20$: INCW PRC_W_OUTMBXREF(R11)
1208 2293 CMPW PRC_W_OUTMBXREF(R11),#1

```

03190004 7E 7C F8 AE 9F DD 8F DD 50 SE DO

52 7E 68 7D EDF0 CF 9E 51 82 9A 7E 51 7D 51 SE DO

04 AB 28 A6 28 A6 C2 68 28 A6 C0 06 50 E8 009B 31 1269 0091 31 126C

01 00CE CB B6 126F 00CE CB B1 1273

: CREATE GETJPI ITEM LIST
: SET BUFFER ADDRESS
: REQUEST OUR PID, SET BUFFER LENGTH
: GET ADDR OF ITEM LIST
: GET PID OF THIS PROCESS

: GET OUR PID
: CLEANUP STACK

: PUSH LOG NAME DESCRIPTOR
: GET ADDRESS OF ASCII FAO STRING
: CONSTRUCT DESCRIPTOR OF STRING
: PUSH DESCRIPTOR ONTO STACK
: GET ADDR OF ARGS
: CONSTRUCT LOGICAL NAME

: FROM PID FOR THIS PROCESS
: POP THE FAO DESCRIPTOR
: GET LOGICAL NAME DESC
: PERMANENTLY REMOVE FROM SCRATCH
: PERMANENTLY REMOVE FROM SCRATCH
: BRANCH IF SUCCESSFUL
: BRANCH IF ERROR DETECTED
: EXIT SUCCESSFULLY

: INCR THE MAILBOX REF COUNT
: DOES MAILBOX ALREADY EXIST?

```

F2 12 1278 2294      BNEQ 10$      ; YES, THEN SKIP MBX CREATION
127A 2295
127A 2296
127A 2297      : GET MAXIMUM MESSAGE SIZE.
127A 2298
25$: 127A 2299      CLRQ  -(SP)      : CREATE THE ITEM LIST
      127C 2300      PUSHAB -8(SP)    : SET THE BUFFER ADDRESS
      127F 2301      PUSHL  #SYIS_MAXBUF@16+4 : SET THE ITEM CODE AND BUFFER SIZE
      1285 2302      MOVL  SP,R0      : GET ADDRESS OF ITEM LIST
      1288 2303      $GETSYIW S,ITMLST=(R0),- : GET MAX BUF SIZE
      1288 2304      EFN=#EXESC_SYSEFN,-
      1288 2305      IOSB=SPWN_Q_IOSB(R6)
      12A0 2306      POPL  R1          : GET THE MAXMSG SIZE
      12A3 2307      ADDL  #3*4,SP     : POP ITEM LIST
      12A6 2308      BLBC  R0,95$     : RETURN IF ERROR
      12A9 2309      MOVZWL SPWN_Q_IOSB(R6),R0 : GET IOSB STATUS
      12AD 2310      BLBC  R0,95$     : RETURN IF ERROR
      12B0 2311      MOVW  R1,PRC_W_OUTMBXSIZ(R11) : SAVE THE MAXMSG SIZE
      12B5 2312
      12B5 2313      :
      12B5 2314      : CREATE THE OUTPUT MAILBOX.
      12B5 2315
      12B5 2316      $CREMBX_S CHAN=PRC_W_OUTMBXCHN(R11),- : CREATE A WRITE MAILBOX
      12B5 2317      MAXMSG=R1,-
      12B5 2318      BUFQUO=R1,-
      12B5 2319      LOGNAM=SPWN_Q_OUTPUT(R6)
      38 50  E9 12CD 2320      BLBC  R0,95$      : BRANCH ON ERROR
      12D0 2321
      12D0 2322      :
      12D0 2323      : SET WRITE ATTENTION AST ON THE MAILBOX.
      12D0 2324
      12D0 2325      $QIOW_S FUNC=#IOS_SETMODE!IOSM_WRTATTN,- : SET ATTENTION AST ON MAILBOX
      12D0 2326      CHAN=PRC_W_OUTMBXCHN(R11),-
      12D0 2327      EFN=#EXESC_SYSEFN,-
      12D0 2328      IOSB=SPWN_Q_IOSB(R6),-
      12D0 2329      P1=W^WRITE_AST,-
      12D0 2330      P2=R11
      12F6 2331      BLBC  R0,95$      : ADDRESS OF AST ROUTINE
      12F9 2332      MOVZWL SPWN_Q_IOSB(R6),R0 : PASS PRC BLOCK ADDRESS
      12FD 2333      BLBC  R0,95$      : BRANCH IF ERROR
      1300 2334      90$: STATUS NORMAL : GET IOSB STATUS
      1307 2335      91$: RSB          : BRANCH IF ERROR
      1308 2336
      1308 2337      : SET NORMAL STATUS
      1308 2338      : RETURN
      1308 2339      :
      1308 2340      : DELETE WRITE MAILBOX AND RETURN WITH STATUS
      130A 2341      95$: PUSHL  R0      : SAVE STATUS
      130D 2342      BSBW  R0      : DELETE ALLOCATED MAILBOX
      1310 2343      POPL  R0      : RESTORE STATUS
      1310 2343      BRB  91$      : EXIT

```

```

1312 2345 .SBTTL DELETE WRITE MAILBOX
1313 2346 :---
1314 2347 :
1315 2348 THIS ROUTINE IS CALLED TO DELETE A WRITE MAILBOX.
1316 2349 :
1317 2350 INPUTS:
1318 2351 :
1319 2352 R11 = ADDRESS OF PRC AREA
1320 2353 :
1321 2354 OUTPUTS:
1322 2355 :
1323 2356 NONE
1324 2357 :---
1325 2358 :
1326 2359 DELETE_OUTMBX:
1327 2360 DECW PRC_W_OUTMBXREF(R11) : DECR THE MAILBOX REF COUNT
1328 2361 BNEQ 90$ : SKIP IF NON-ZERO
1329 2362 $DASSGN_S CHAN=PRC W OUTMBXCHN(R11) : DELETE OUTPUT MAILBOX
1330 2363 CLRW PRC_W_OUTMBXCHN(R11) : CLEAR MAILBOX CHANNEL NUMBER
1331 2364 90$: RSB :

```

```

1329 2366 .SBTTL WRITE REQUEST AST FROM A SUBPROCESS
1329 2367 ---
1329 2368
1329 2369 THIS AST ROUTINE HANDLES A WRITE REQUEST FROM A SUBPROCESS
1329 2370 THROUGH THE WRITE MAILBOX ASSOCIATED WITH THIS PROCESS.
1329 2371
1329 2372 INPUTS:
1329 2373
1329 2374 4(AP) = ADDRESS OF PRC AREA
1329 2375
1329 2376 OUTPUTS:
1329 2377
1329 2378 NONE
1329 2379 ---
1329 2380
1329 2381 WRITE_AST:
1329 2382 .WORD ^M<R2,R3,R4,R5,R6,R11>
1329 2383
1329 2384 MOVL 4(AP),R11 ; GET ADDRESS OF CLI PROCESS WORK AR
1329 2385
1329 2386
1329 2387 IF HANGUP AST IS PENDING, THEN WAKE UP SO THAT WE CAN TERMINATE.
1329 2388
1329 2389 BBS #PRC_V_HANGUP,PRC_W_FLAGS(R11),80$ ; WAKE UP IF HANGUP PENDING
1329 2390
1329 2391
1329 2392 RE-ENABLE WRITE ATTENTION AST.
1329 2393
1329 2394 CLRQ -(SP) ; ALLOCATE AN IOSB
1329 2395 MOVL SP,R4
1329 2396 $QIOW_S FUNC=#IOS SETMODE!IOSM WRTATTN,- ; RE-ENABLE AST
1329 2397 CHAN=PRC_W_OUTMBXCHN(R11),-
1329 2398 EFN=#EXESC_SYSEFN,-
1329 2399 IOSB=(R4),-
1329 2400 P1=WRITE_AST,- ; ADDRESS OF AST ROUTINE
1329 2401 P2=R11 ; ADDRESS OF PRC DATA STRUCTURE
1329 2402
1329 2403
1329 2404 ALLOCATE A RECORD BUFFER AND IOSB.
1329 2405
1329 2406 MOVZWL PRC_W_OUTMBXSIZ(R11),R5 ; GET SIZE OF RECORD BUFFER
1329 2407 SUBL R5,SP ; ALLOCATE RECORD BUFFER
1329 2408 MOVL SP,R2 ; GET THE BUFFER ADDRESS
1329 2409
1329 2410
1329 2411 GET RECORD TO BE WRITTEN FROM THE MAILBOX. IF THERE IS MORE THAN
1329 2412 ONE MESSAGE IN THE MAILBOX, LOOP UNTIL WE'VE READ AND WRITTEN THEM ALL.
1329 2413
1329 2414 IOS: $QIOW_S FUNC=#IOS READVBLK!IOSM NOW,- ; READ THE ATTACH MAILBOX
1329 2415 CHAN=PRC_W_OUTMBXCHN(R11),-
1329 2416 EFN=#EXESC_SYSEFN,-
1329 2417 IOSB=(R4),-
1329 2418 P1=(R2),- ; ADDRESS OF BUFFER
1329 2419 P2=R5 ; SIZE OF BUFFER
1329 2420 BLBC R0,30$ ; BRANCH IF DSW ERROR
1329 2421 BLBC (R4),20$ ; BRANCH IF I/O ERROR
1329 2422 CVTWL 2(R4),R1 ; GET LENGTH OF MESSAGE

```

SPAWN
V04-000

- MULTI-PROCESSING COMMANDS
WRITE REQUEST AST FROM A SUBPROCESS

M 15

16-SEP-1984 00:17:05
4-SEP-1984 23:43:20

VAX/VMS Macro V04-00
[DCL.SRC]SPAWN.MAR;1

Page 54
(23)

EC68'	30	1395	2423	BSBW	DCL\$SPAWNOUT	: WRITE THE MESSAGE
CE	11	1398	2424	BRB	108	: LOOP UNTIL MAILBOX CLEANED OUT
		139A	2425			
0870 BF	64	B1	139A	2426	208: CMPW	(R4),#SS\$_ENDOFFILE
	C7	12	139F	2427	BNEQ	108
04	A4	D5	13A1	2428	TSTL	4(R4)
	C2	12	13A4	2429	BNEQ	108
			13A6	2430	STATUS	NORMAL
	04		13AD	2431	308: RET	
			13AE	2432		
			13AE	2433	808: SWAKE_S	
	04		13B9	2434	RET	

: WAS ERROR EOF?
: NO, DISREGARD AND READ AGAIN
: WAS PID SPECIFIED?
: YES, THEN KEEP READING TO EMPTY
: SET NORMAL STATUS
: RETURN
: WAKE UP CURRENT PROCESS

```

13BA 2436 .SBTTL READ AST FROM READING ATTACH REQUEST RESPONSE
13BA 2437 :---
13BA 2438 :
13BA 2439 : THIS AST ROUTINE IS CALLED UPON THE COMPLETION OF READING THE
13BA 2440 : ATTACH REQUEST RESPONSE. IT HANDLES THE PROCESSING OF THE
13BA 2441 : DETACHED FLAG. IT IS DOWN AT AST LEVEL TO INSURE SYNCHRONIZATION
13BA 2442 : BETWEEN THE CURRENT PROCESS AND THE PROCESS TO BE ATTACHED.
13BA 2443 : THE ADDRESS OF A QUAD WORD IS PASSED AS THE AST PARAMETER. THE QUAD WORD
13BA 2444 : CONTAINS THE INPUT BUFFER FROM THE READ COMMAND (1ST LONG WORD) AND
13BA 2445 : THE ADDRESS OF THE PRC TABLE AREA (2ND LONG WORD).
13BA 2446 :
13BA 2447 : INPUTS:
13BA 2448 :
13BA 2449 : 4(SP) = AST PARAMETER
13BA 2450 :
13BA 2451 : OUTPUTS:
13BA 2452 :
13BA 2453 : PRC V DETACHED BIT IN PRC_W_FLAGS SET IF INPUT BUFFER = 1
13BA 2454 : CLEARED OTHERWISE
13BA 2455 :
13BA 2456 :---
13BA 2457 :
0804 13BA 2458 READ_AST:
13BA 2459 .WORD *M(R2,R11)
13BC 2460 :
52 04 AC D0 13BC 2461 MOVL 4(AP),R2 ; GET ADDR OF QUAD WORD BUFFER
5B 04 A2 D0 13C0 2462 MOVL 4(R2),R11 ; GET ADDR. OF CLI PROCESS WORK AREA
05 62 E9 13C4 2463 BLBC (R2),10$ ; DON'T SET DETACHED BIT IF ATTACH REFUSED
13C7 2464 SETBIT PRC_V_DETACHED,PRC_W_FLAGS(R11) ; MARK CURRENT PROC. DETACHED
13CC 2465
04 13CC 2466 10$: RET

```

```

13CD 2468 .SBTTL ATTACH REQUEST AST FROM ANOTHER PROCESS
13CD 2469
13CD 2470
13CD 2471 THIS AST ROUTINE HANDLES AN ATTACH REQUEST FROM ANOTHER PROCESS
13CD 2472 THROUGH THE ATTACH MAILBOX ASSOCIATED WITH THIS PROCESS.
13CD 2473
13CD 2474 INPUTS:
13CD 2475
13CD 2476 4(AP) = ADDRESS OF PRC AREA
13CD 2477
13CD 2478 OUTPUTS:
13CD 2479
13CD 2480 NONE
13CD 2481
13CD 2482
13CD 2483 ATTACH_AST:
13CD 2484 .WORD ^M<R2,R3,R4,R5,R6,R11>
13CF 2485
13CF 2486 MOVL 4(AP),R11 ; GET ADDRESS OF CLI PROCESS WORK AREA
13D3 2487 MOVAB -ATTMBX_MAXMSG(SP),R6 ; ALLOCATE RECORD BUFFER
13D7 2488 MOVL R6,SP ; REMOVE IT FROM THE STACK
13DA 2489
13DA 2490
13DA 2491 IF HANGUP AST IS PENDING, THEN WAKEUP PROCESS SO WE CAN TERMINATE
13DA 2492
13DA 2493 BBS #PRC_V_HANGUP,PRC_W_FLAGS(R11),80$ ; WAKE UP IF HANGUP PENDING
13DF 2494
13DF 2495
13DF 2496 GET THE ATTACH REQUEST MESSAGE FROM THE MAILBOX. IF THERE IS MORE THAN
13DF 2497 ONE MESSAGE IN THE MAILBOX, ONLY HONOR THE LAST ONE.
13DF 2498
13DF 2499 CLRL R3 ; ASSUME NO MESSAGE READ
13E1 2500 CLRQ -(SP) ; ALLOCATE AN IOSB
13E3 2501 MOVL SP,R2 ; POINT TO IT
13E6 2502
13E6 2503 10$: SQIOW_S FUNC=#IOS READVBLK!IOSM_NOW,- ; READ THE ATTACH MAILBOX
13E6 2504 CHAN=PRC @ ATTMBX(R11),-
13E6 2505 EFN=#EXESC_SYSEFN,-
13E6 2506 IOSB=(R2),-
13E6 2507 P1=(R6),P2=#ATTMBX_MAXMSG
13E6 2508 BLBC R0,20$ ; BRANCH IF ERROR
13E6 2509 BLBC (R2),20$ ; BRANCH IF ERROR (EOF)
13E6 2510 CVTWL 2(R2),R3 ; SAVE LENGTH OF LAST MESSAGE
13E6 2511 BRB 10$ ; LOOP UNTIL MAILBOX CLEANED OUT
13E6 2512 TSTL R3 ; ANY MESSAGE READ?
13E6 2513 BEQL 90$ ; IF NOT, IGNORE ATTENTION AST
13E6 2514 CLRL R4 ; ASSUME THE ANSWER IS 'NO'
13E6 2515 BBC #PRC_V_DETACHED,PRC_W_FLAGS(R11),30$ ; IF NOT DETACHED, SAY NO
13E6 2516 INCL R4 ; ELSE, RESPOND WITH YES
13E6 2517 MOVL SP,R2 ; RESTORE ADDRESS OF IOSB
13E6 2518 MOVL R4,(R6) ; MOVE RECORD INTO RECORD BUFFER
13E6 2519 SQIOW_S FUNC=#IOS WRITEVBLK,- ; WRITE YES/NO RESPONSE TO MAILBOX
13E6 2520 CHAN=PRC @ ATTMBX(R11),-
13E6 2521 EFN=#EXESC_SYSEFN,-
13E6 2522 IOSB=(R2),-
13E6 2523 P1=(R6),P2=#4 ; ONLY A LONGWORD
13E6 2524 BLBC R0,90$ ; BRANCH IF ERROR DETECTED

```

SPAWN
V04-000

- MULTI-PROCESSING COMMANDS
ATTACH REQUEST AST FROM ANOTHER PROCESS

C 16

16-SEP-1984 00:17:05 VAX/VMS Macro V04-00
4-SEP-1984 23:43:20 [DCL.SRC]SPAWN.MAR;1

Page 57
(25)

13 62	E9	144A	2525	BLDC	(R2), 908	: BRANCH IF ERROR DETECTED
10 54	E9	144D	2526	BLDC	R4, 908	: IF ANSWER WAS NO, THEN EXIT
		1450	2527	CLRBIT	PRC_V_DETACHED, PRC_W_FLAGS(R11)	: MARK NO LONGER DETACHED
		1455	2528	808:	SWAKE_S	: WAKE UP CURRENT PROCESS
04		1460	2529	908:	RET	

```

1461 2531 .SBTTL SUBPROCESS TERMINATION AST ROUTINE
1461 2532 ---
1461 2533
1461 2534 THIS AST IS CALLED WHEN A MESSAGE IS WRITTEN INTO THE TERMINATION
1461 2535 MAILBOX, INDICATING THAT A SUBPROCESS HAS GONE AWAY.
1461 2536
1461 2537 INPUTS:
1461 2538
1461 2539 4(AP) = ADDRESS OF TMBX AREA
1461 2540
1461 2541 OUTPUTS:
1461 2542
1461 2543 NONE
1461 2544 ---
1461 2545
000000D3 1461 2546 LGIS_FACILITY = 211 ; LOGINOUT FACILITY CODE
1461 2547
09EC 1461 2548 TERMINATION AST:
1461 2549 .WORD *M<R2,R3,R5,R6,R7,R8,R11>
1463 2550
1463 2551 :
1463 2552 : GET THE ADDRESS OF THE TMBX DATA STRUCTURE FROM THE AST ARGUMENT LIST.
1463 2553 : LOOK IN IT FOR THE ADDRESS OF THE PRC DATA STRUCTURE AND THE TERMINATION
1463 2554 : MAILBOX CHANNEL NUMBER.
1463 2555 :
1463 2556 :
57 04 AC DO 1463 2556 MOVL 4(AP),R7 ; GET ADDRESS OF TMBX BLOCK
5B 0C A7 DO 1467 2557 MOVL TMBX_L_PRC(R7),R11 ; GET ADDRESS OF CLI PROCESS WORK AR
1468 2558
1468 2559 :
1468 2560 : READ THE TERMINATION MAILBOX. IF NO MESSAGES ARE LEFT, THEN WE ARE READY TO
1468 2561 : RESET OR DELETE THE MAILBOX AND RETURN FROM THE AST.
1468 2562 :
1468 2563 :
SE A4 AE 9E 1468 2563 MOVAB -8-ACCSC_TERMLEN(SP),SP ; ALLOCATE BUFFER SPACE FOR
5B 5E DO 146F 2564 10$: MOVL SP,R8 ; THE IOSB + RECORD BUFFER
1472 2565 $QIOW_S FUNC=#IOS_READVBLK!IOSM_NOW,- ; READ THE TERMINATION MAILBOX
1472 2566 CHAN=TMBX_W_CHANNEL(R7),-
1472 2567 EFN=#EXESC_SYSEFN,-
1472 2568 IOSB=(R8),-
1472 2569 P1=8(R8),-
1472 2570 P2=#ACCSC_TERMLEN
03 50 E9 1499 2571 BLBC R0,15$ ; BRANCH IF ERROR
03 68 E8 149C 2572 BLBS (R8),20$ ; BRANCH IF MESSAGE READ
00E2 31 149F 2573 15$: BRW 200$ ; MAILBOX CLEANED OUT - DONE
14A2 2574
14A2 2575 :
14A2 2576 : SEARCH THE LIST OF OUTSTANDING SUBPROCESS CONTEXT BLOCKS LOOKING FOR
14A2 2577 : A MATCH WITH THE SUBPROCESS WHICH JUST TERMINATED.
14A2 2578 :
14A2 2579 :
55 00C0 CB 9E 14A2 2580 20$: ASSUME SPWN_L_LINK EQ 0
56 65 DO 14A7 2581 25$: MOVAB PRC_C_SPWN(R11),R5 ; GET ADDRESS OF SPWN BLOCK LISTHEAD
C3 13 14AA 2582 MOV L SPWN_C_LINK(R5),R6 ; GET ADDRESS OF NEXT ENTRY IN LIST
10 A8 D1 14AC 2583 BEQL 10$ ; IF NOT IN LIST, IGNORE MESSAGE
40 A6 14AF 2584 CMPL ACCBL PID+8(R8),- ; DOES PID MATCH?
SPWN_C_SUBPID(R6)
14B1 2585 BEQL 30$ ; BRANCH IF SO
55 56 DO 14B3 2586 MOVL R6,R5 ; SKIP TO NEXT ENTRY IN LIST
EF 11 14B6 2587 BRB 25$ ; KEEP LOOPING

```

```

1488 2588
1488 2589
1488 2590 : THE PROCESS WHICH JUST TERMINATED HAS BEEN FOUND IN THE LIST OF
1488 2591 : CURRENTLY ACTIVE SPAWNED PROCESSES. TAKE ANY TERMINATION ACTIONS WHICH
1488 2592 : ARE APPROPRIATE.
1488 2593
1488 2594 : SET THE TERMINATION STATUS IN THE SPWN BLOCK.
1488 2595
OC A8 DO 1488 2596 50$: MOVL ACCBL_FINALSTS+8(RB),- : SAVE THE TERMINATION STATUS
44 A6 04 12 1488 2597 : SPWN_L_STATUS(R6)
44 A6 01 DO 148D 2598 : BNEQ 50$ : BRANCH IF STATUS IS NON-ZERO
148F 2599 : MOVL #1,SPWN_L_STATUS(R6) : IF 'NO STATUS', RETURN SUCCESS
14C3 2600
14C3 2601
14C3 2602 : GET CURRENT IMAGE COUNT. IF NOT THE SAME AS THE IMAGE COUNT AT THE TIME
14C3 2603 : OF SUBPROCESS CREATION, THEN DO NOT EVEN CONSIDER QUEUEING A TERMINATION,
14C3 2604 : SETTING A TERMINATION EVENT FLAG, OR SETTING TERMINATION STATUS.
14C3 2605
7E 7C 14C3 2606 50$: CLRQ -(SP) : CREATE GETJPI ITEM LIST
FB AE 9F 14C5 2607 : PUSHAB -2*4(SP) : SET BUFFER ADDRESS
041A0004 8F DD 14C8 2608 : PUSHL #JPIS_IMAGECOUNT@16+4 : REQUEST IMAGE COUNT, SET BUFFER LE
50 SE DO 14CE 2609 : MOVL SP,R0 : SET ADDRESS OF ITEM LIST
14D1 2610 : $GETJPIW S ITMLST=(R0),- : GET PROCESS IMAGE COUNT
14D1 2611 : IOSB=(RB),-
14D1 2612 : EFN=#EXES&_SYSEFN
SE 51 8ED0 14E8 2613 : POPL R1 : GET THE CURRENT IMAGE COUNT
5C A6 0C CO 14EB 2614 : ADDL #3*4,SP : CLEANUP STACK
5C A6 51 D1 14EE 2615 : CMPL R1,SPWN_L_IMAGCNT(R6) : SAME IMAGE AS ISSUED SPAWN?
37 12 14F2 2616 : BNEQ 80$ : NO, THEN DO NOT QUEUE AST OR SET E
14F4 2617
14F4 2618
14F4 2619 : IF USER REQUESTED THAT THE TERMINATION STATUS BE RETURNED, THEN TRY TO RETURN
14F4 2620 : IT NOW.
50 54 A6 DO 14F4 2621 : MOVL SPWN_L_STSADR(R6),R0 : SHOULD STATUS BE RETURNED TO CALLE
OD 13 14F8 2622 : BEQL 60$ : BRANCH IF NOT
14FA 2623 : IFNOWRT #4,@SPWN_L_STSADR(R6),60$,- : SKIP IF NOT WRITABLE
14FA 2624 : SPWN_B_ACMODE(R6)
44 A6 DO 1502 2625 : MOVL SPWN_L_STATUS(R6),- : RETURN STATUS TO CALLER
54 B6 1505 2626 : @SPWN_L_STSADR(R6)
1507 2627
1507 2628
1507 2629 : IF A TERMINATION AST WAS REQUESTED, THEN QUEUE IT UP NOW.
1507 2630
4C A6 D5 1507 2631 60$: TSTL SPWN_L_ASTADR(R6) : AST ROUTINE REQUESTED?
10 13 150A 2632 : BEQL 70$ : BRANCH IF NOT
150C 2633 : $DCLAST_S ASTADR=@SPWN_L_ASTADR(R6),- : QUEUE THE AST
150C 2634 : ASTPRM=SPWN_L_ASTPRM(R6),-
150C 2635 : ACMODE=SPWN_B_ACMODE(R6)
151C 2636
151C 2637
151C 2638 : IF A TERMINATION EF WAS REQUESTED, THEN SET IT NOW.
151C 2639
OF A6 95 151C 2640 70$: TSTB SPWN_B_EFN(R6) : EVENT FLAG REQUESTED?
OA 19 151F 2641 : BLSS 80$ : BRANCH IF NOT
1521 2642 : $SETEF_S EFN=SPWN_B_EFN(R6) : SET THE EVENT FLAG
152B 2643
152B 2644 :

```

```

152B 2645 : PREVENT THE PARENT FROM HANGING TRYING TO SEND ADDITIONAL MESSAGES TO THE
152B 2646 : CONTEXT MAILBOX BY CANCELING AND DEASSIGNING THE CHANNEL TO THAT MAILBOX.
152B 2647 :
152B 2648 808: SCANCEL_S CHAN=SPWN_W_CHAN(R6) : STOP ANY CURRENT WRITES TO MAILBOX
1536 2649 SDASSGN_S CHAN=SPWN_W_CHAN(R6) : PREVENT FURTHER WRITES TO MAILBOX
OA A6 B4 1541 2650 CLRW SPWN_W_CHAN(R6) : INDICATE CHANNEL WAS "REMOVED"
1544 2651 :
1544 2652 : IF /NOTIFY SPECIFIED, THEN BROADCAST NOTIFICATION MESSAGE.
1544 2653 :
1544 2654 908: BBC #SPWN_V_NOTIFY - : SKIP IF NOT /NOTIFY
03 08 E1 1544 2655 SPWN_W_FLAGS(R6),958 :
03 0C A6 1546 2656 BSBW NOTIFY_MESSAGE : BROADCAST NOTIFICATION MESSAGE
0067 30 1549 2657 :
154C 2658 :
154C 2659 : DECREMENT TERMINATION MAILBOX REFERENCE COUNT.
154C 2660 :
154C 2661 958: DECB TMBX_B_REFS(R7) : DECREMENT REFERENCE COUNT TO MAILB
154F 2662 :
154F 2663 : DELETE THE WRITE MAILBOX IF ALL THROUGH WITH IT.
154F 2664 :
154F 2665 :
154F 2666 :
03 08 E0 154F 2667 BBS #SPWN_V_OUTPUT - : SKIP IF NO MAILBOX
03 0C A6 1551 2668 SPWN_W_FLAGS(R6),978 :
FD8B 30 1554 2669 BSBW DELETE_OUTMBX : UPDATE OUTPUT MBX USAGE
1557 2670 :
1557 2671 :
1557 2672 : IF SUBPROCESS WAS CREATED /WAIT, THEN ASSUME THAT THE SUBPROCESS THAT JUST
1557 2673 : DIED WAS ATTACHED TO THE TERMINAL AND IS NOW LOGGING ITSELF OUT. ALSO,
1557 2674 : ASSUME THAT THE PARENT PROCESS WAS DETACHED FROM THE TERMINAL AND SHOULD
1557 2675 : NOW BE AWAKENED TO RECEIVE CONTROL.
1557 2676 :
1557 2677 : NOTE THAT THESE ASSUMPTIONS ARE NOT VALID IN THE CASE OF AN "ATTACHED"
1557 2678 : COUSIN DELETING ITS "DETACHED" COUSIN.
1557 2679 :
1557 2680 : ALSO MARK THE SUBPROCESS INACTIVE SO THAT THE PARENT WILL DELETE THIS
1557 2681 : SPWN BLOCK.
1557 2682 :
1557 2683 978: BBC #SPWN_V_WAIT - : SKIP IF /NOWAIT
18 02 E1 1557 2684 SPWN_W_FLAGS(R6),1008 :
1559 2685 CLRBTT PRC_V_DETACHED,PRC_W_FLAGS(R11) : MARK NO LONGER DETACHED
155C 2686 CLRBTT SPWN_V_ACTIVE,SPWN_W_FLAGS(R6) : MARK SUBPROCESS INACTIVE
1561 2687 SWAKE_S : WAKE UP CURRENT PROCESS
FEFB 31 1571 2688 BRW 108 : GET NEXT MESSAGE FROM MAILBOX
1574 2689 :
1574 2690 : IF THE SUBPROCESS WAS CREATED /NOWAIT, THEN REMOVE THE SPWN BLOCK FROM THE
1574 2691 : LIST OF OUTSTANDING PROCESSES, AND DEALLOCATE IT.
1574 2692 :
1574 2693 :
1574 2694 1008: ASSUME SPWN_L_LINK EQ 0 :
1574 2695 MOVL (R6),(R5) : REMOVE FROM LINKED LIST NOW THAT
1577 2696 : THE PROCESS IS NO LONGER ACTIVE
50 56 DO 1577 2697 MOVL R6,R0 : SET ADDRESS OF BLOCK
51 04 A6 3C 157A 2698 MOVZWL SPWN_W_SIZE(R6),R1 : SET LENGTH OF BLOCK
EA7F 30 157E 2699 BSBW DCL$DEADYNMEM : DEALLOCATE SPWN BLOCK
FEFB 31 1581 2700 BRW 108 : LOOP UNTIL MAILBOX CLEANED OUT
1584 2701 :

```

```

1584 2702 :
1584 2703 : IF REF COUNT FOR THIS MAILBOX IS NOW ZERO, THEN DEASSIGN IT AND REMOVE
1584 2704 : THE TMBX BLOCK FROM THE LINKED LIST. OTHERWISE, RE-ENABLE WRITE ATTENTION
1584 2705 : AST ON THE MAILBOX.
1584 2706 :
08 A7 95 1584 2707 200$: TSTB TMBX_B_REFS(R7) : TEST REFERENCE COUNT TO MAILBOX
05 12 1587 2708 : BNEQ 210$ : BRANCH IF STILL OUTSTANDING USES
F873 30 1589 2709 : BSBW DELETE_TMBX : DELETE TERMINATION MAILBOX AND TMB
24 11 158C 2710 : BRB 220$ : DO NOT BOTHER RESETTING THE AST
158E 2711 210$: $QIOW,S FUNC=#IOS_SETMODE!IOSM WRTATTN,- : RESET ATTENTION AST ON MAILBOX
158E 2712 : CHAN=TMBX_W_CHANNEL(R7),-
158E 2713 : IOSB=(R8),-
158E 2714 : EFN=#EXESC SYSEFN,-
158E 2715 : P1=TERMINATION_AST,-
158E 2716 : P2=R7
04 1582 2717 220$: RET : ADDRESS OF AST ROUTINE
1583 2718 : PASS ADDRESS OF TMBX BLOCK

```

```

15B3 2720 .SBTTL BROADCAST NOTIFICATION MESSAGE
15B3 2721
15B3 2722
15B3 2723 THIS ROUTINE BROADCASTS A MESSAGE INDICATING THAT A /NOWAIT SUBPROCESS HAS
15B3 2724 TERMINATED.
15B3 2725
15B3 2726 INPUTS:
15B3 2727
15B3 2728 NONE
15B3 2729
15B3 2730 OUTPUTS:
15B3 2731
15B3 2732 NONE
15B3 2733
15B3 2734
15B3 2735 NOTIFY_MESSAGE:
15B3 2736 SUBL #NOTIFY_LEN,SP : ALLOCATE BUFFER ON THE STACK
15B3 2737 MOVL SP,-(SP) : BUILD BUFFER DESCRIPTOR
15B3 2738 MOVL #NOTIFY_LEN,-(SP)
15B3 2739 MOVL SP,R3 : SAVE ADDRESS OF DESCRIPTOR
15B3 2740 MOVAB SPWN_T_PROCESS(R6),R2 : GET PROCESS NAME
15B3 2741 MOVAB NOTIFY_MSG,R1 : GET ADDRESS OF ASCII MSG
15B3 2742 MOVZBL (R1)+,R0 : GET LENGTH OF MSG
15B3 2743 MOVQ R0,-(SP) : PUSH DESCRIPTOR ON STACK
15B3 2744 MOVL SP,R1 : SAVE ADDRESS OF DESCRIPTOR
15B3 2745 $FAO_S CTRSTR=(R1),- : NOTIFICATION MESSAGE
15B3 2746 : ADDR OF RESULT LENGTH
15B3 2747 : ADDR OF BUFFER DESC
15B3 2748 P1=R2 : ASCII SUBPROCESS NAME ADDR
15E1 2749 MOVL #CTLSAG CLIDATA+PPDST_INPDVI,R1 : GET ADDRESS OF ASCII DEVICE NAME
15E8 2750 MOVZBL (R1)+,R0 : GET LENGTH OF DEVICE
15EB 2751 MOVQ R0,(SP) : GET DESCRIPTOR ON STACK
15EE 2752 MOVL SP,R2 : SAVE ADDRESS OF DESCRIPTOR
15F1 2753 CLRQ -(SP) : ALLOCATE AN IOSB
15F3 2754 MOVL (SP),R0
15F6 2755
15F6 2756 $BRKTHRUW_S MSGBUF=(R3),- : BROADCAST THE MESSAGE
15F6 2757 SENDTO=(R2),-
15F6 2758 SNDTYP=#BRK$C_DEVICE,-
15F6 2759 REQID=#BRK$C_DCL,-
15F6 2760 EFN=#31,-
15F6 2761 IOSB=(R0)
1611 2762
1611 2763 ADDL #8+8+8+NOTIFY_LEN,SP : RESTORE STACK
1618 2764 RSB
1619 2765
1619 2766 .END

```

[illegible]

SPAWN
Symbol table

- MULTI-PROCESSING COMMANDS

J 16

16-SEP-1984 00:17:05 VAX/VMS Macro V04-00
4-SEP-1984 23:43:20 [DCL.SRC]SPAWN.MAR;1

Page 64
(27)

DCL\$SPAWN	00000098	RG	02	LNMHSHSC_BUCKET	= 0000000C		
DCL\$SPAWN2	0000025F	RG	02	LNMHSHSL_MASK	= 00000000		
DCL\$SPAWNOUT	*****	X	02	LNMTSHL_BYTESLM	= 0000001D		
DEALLOC_SPWN	00000E19	R	02	LNMTSHL_CHILD	= 00000011		
DELETE_ATTMBX	00000D62	R	02	LNMTSHL_NAME	= 00000009		
DELETE_OUTMBX	00001312	R	02	LNMTSHL_PARENT	= 0000000D		
DELETE_TMBX	00000DFF	R	02	LNMTSHL_SIBLING	= 00000015		
DEVSV_REC	= 00000000			LNMXSB_FLAGS	= 00000000		
DEVSV_TRM	= 00000002			LNMXST_XLATION	= 00000004		
DIBSW_UNIT	= 0000000C			LNMXSV_XEND	= 00000002		
DVIS_DEVCLASS	= 00000004			LOG	00000065	R	02
DVIS_DEVNAM	= 00000020			LOGINOUT	00000000	R	02
ENT_R_MAX_PROMPT	= 00000020			NAMSB_BID	= 00000000		
EXESC_SYSEFN	*****	X	02	NAMSB_BLN	= 00000001		
FABSB_BID	= 00000000			NAMSB_DEV	= 00000039		
FABSB_BLN	= 00000001			NAMSB_ESL	= 0000000B		
FABSB_DNS	= 00000035			NAMSB_ESS	= 0000000A		
FABSB_FNS	= 00000034			NAMSB_NOP	= 00000008		
FABSC_BID	= 00000003			NAMSC_BID	= 00000002		
FABSC_BLN	= 00000050			NAMSC_BLN	= 00000060		
FABSL_DEV	= 00000040			NAMSC_MAXRSS	= 000000FF		
FABSL_DNA	= 00000030			NAMSL_DEV	= 00000044		
FABSL_FNA	= 0000002C			NAMSL_ESA	= 0000000C		
FABSL_FOP	= 00000004			NAMSL_FNB	= 00000034		
FABSL_NAM	= 00000028			NAMSM_NOCONCEAL	= 00000010		
FABSV_PPF	= 00000012			NAMSV_PPF	= 00000010		
GET_DEVICE	000011C6	R	02	NL	0000005B	R	02
IOSM_NOW	= 00000040			NOTIFY_LEN	= 0000002C		
IOSM_TT_PROCESS	= 00002000			NOTIFY_MESSAGE	000015B3	R	02
IOSM_WRTATTN	= 00000100			NOTIFY_MSG	00000079	R	02
IOS_READVBLK	= 00000031			OUTPUT_NAME	0000002B	R	02
IOS_SENSEMODE	= 00000027			PPDSB_RPROCS	0000001C		
IOS_SETMODE	= 00000023			PPDSC_LENGTH	00000168		
IOS_WRITEOF	= 00000028			PPDSK_LENGTH	00000168		
IOS_WRITEVBLK	= 00000030			PPDSL_INPDEV	00000044		
JPIS_ASTLM	= 00000409			PPDSL_LGI	00000014		
JPIS_BIOLM	= 00000310			PPDSL_LSTSTATUS	00000018		
JPIS_DFWSCNT	= 00000403			PPDSL_OUTDEV	00000064		
JPIS_DIOLM	= 00000313			PPDSL_PRC	00000008		
JPIS_IMAGECOUNT	= 0000041A			PPDSQ_CLIREG	00000004		
JPIS_MASTER_PID	= 00000325			PPDSQ_CLISYMTBL	0000000C		
JPIS_PID	= 00000319			PPDST_FILENAME	00000068		
JPIS_PRCNAM	= 0000031C			PPDST_INPDVI	00000028		
JPIS_PRI8	= 00000309			PPDST_OUTDVI	00000048		
JPIS_PROCPRIV	= 00000204			PPDSW_FLAGS	00000002		
JPIS_USERNAME	= 00000202			PPDSW_INPCHAN	0000001E		
JPIS_WSEXTENT	= 00000416			PPDSW_INPDID	0000003E		
JPIS_USQUOTA	= 00000402			PPDSW_INPFID	00000038		
LGIS_FACILITY	= 000000D3			PPDSW_INPFI	00000020		
LNMSC_NAMLENGTH	= 000000FF			PPDSW_INPISI	00000022		
LNMB\$B_ACMODE	= 0000000B			PPDSW_OUTDID	0000005E		
LNMB\$B_FLAGS	= 00000010			PPDSW_OUTFID	00000058		
LNMB\$B_TABLE	= 0000000C			PPDSW_OUTIFI	00000024		
LNMB\$M_TABLE	= 00000008			PPDSW_OUTISI	00000026		
LNMB\$T_NAME	= 00000011			PPDSW_SIZE	00000000		
LNMB\$V_CONFINE	= 00000001			PQL\$_ASTLM	= 00000001		
LNMB\$V_TABLE	= 00000003			PQL\$_BIOLM	= 00000002		

SPAWN
Symbol table

- MULTI-PROCESSING COMMANDS

K 16

16-SEP-1984 00:17:05 VAX/VMS Macro V04-00
4-SEP-1984 23:43:20 [DCL.SRC]SPAWN.MAR;1

Page 65
(27)

```

PQLS_DIOLM      = 00000005
PQLS_LISTEND    = 00000000
PQLS_USDEFAULT  = 00000008
PQLS_USEXTENT   = 00000000
PQLS_USQUOTA    = 0000000A
PRCSM_CLISPEC   = 00001000
PRCSM_INTER     = 00000400
PRCNAM_NAME     = 00000014
PRC_B_CONTINUE  = 000000F3
PRC_B_DEFRADIX  = 000000AE
PRC_B_EXMDEPADR = 000000AD
PRC_B_EXMDEPWID = 000000AC
PRC_B_EXONLYL   = 0000012D
PRC_B_FLAGS2    = 000000AF
PRC_B_IMGFLAG   = 00000078
PRC_B_OUTFLAGS  = 0000012C
PRC_B_PROMPTLEN = 000000F0
PRC_C_LENGTH    = 00000534
PRC_G_COMMANDS  = 00000133
PRC_G_PROMPT    = 000000F4
PRC_K_HEX       = 00000000
PRC_K_LENGTH    = 00000534
PRC_L_CURRKEY   = 00000048
PRC_L_EXMDEPADR = 000000A8
PRC_L_EXTARG    = 00000094
PRC_L_EXTBLK    = 0000008C
PRC_L_EXTCOD    = 0000009C
PRC_L_EXTHND    = 00000090
PRC_L_EXTPRM    = 00000098
PRC_L_IDFLNK    = 0000008C
PRC_L_IMGACTSTS = 00000080
PRC_L_INDCLOCK  = 0000007C
PRC_L_INDEPTH   = 0000005C
PRC_L_INDFAB    = 0000001C
PRC_L_INDINPRAB = 00000014
PRC_L_INDOUSTRAB = 00000018
PRC_L_INPRAB    = 00000008
PRC_L_LASTKEY   = 0000004C
PRC_L_LSTSTATUS = 000000B0
PRC_L_ONCTLY    = 000000B8
PRC_L_ONERROR   = 0000006C
PRC_L_OUTOFBAND = 000000B4
PRC_L_OUTRAB    = 0000000C
PRC_L_OUTRABCTX = 00000118
PRC_L_PPFLIST   = 00000070
PRC_L_RECALLPTR = 0000012F
PRC_L_RESTART   = 00000058
PRC_L_SAVAP     = 00000000
PRC_L_SAVFP     = 00000004
PRC_L_SEVERITY  = 00000050
PRC_L_SPWN      = 000000C0
PRC_L_STACKLM   = 000000A4
PRC_L_STACKPT   = 000000A0
PRC_L_STATUS    = 00000054
PRC_L_STS       = 00000084
PRC_L_STV       = 00000088
PRC_L_SYMBOL    = 00000060

```

R 02

```

PRC_L_TMBX      = 00000074
PRC_L_TRMLIST   = 00000010
PRC_Q_ALLOCREG  = 00000020
PRC_Q_COMMAND   = 000000E0
PRC_Q_FLUSHTIME = 000000D0
PRC_Q_GLOBAL    = 00000028
PRC_Q_IMAGENAME = 000000D8
PRC_Q_KEYPAD    = 00000040
PRC_Q_LABEL     = 00000030
PRC_Q_LOCAL     = 00000038
PRC_Q_SAVEPRIV  = 000000E8
PRC_T_OUTDVI    = 0000011C
PRC_V_DETACHED  = 0000000F
PRC_V_HANGUP    = 0000000C
PRC_V_MODE      = 00000006
PRC_V_VERIFY    = 00000007
PRC_V_VERIMAGE  = 00000007
PRC_V_YLEVEL    = 0000000B
PRC_W_ASTIOSB   = 000000C6
PRC_W_ASTRETN   = 000000C8
PRC_W_ASTSTATUS = 000000C4
PRC_W_ATMBX     = 0000007A
PRC_W_FLAGS     = 00000068
PRC_W_INPCHAN   = 00000064
PRC_W_ONLEVEL   = 0000006A
PRC_W_OUTIFI    = 00000114
PRC_W_OUTISI    = 00000116
PRC_W_OUTMBXCHN = 000000CA
PRC_W_OUTMBXREF = 000000CE
PRC_W_OUTMBXSIZ = 000000CC
PRC_W_PMPTCTRL  = 000000F1
PRC_W_WAITIOSB  = 00000066
PSLSC_EXEC      = 00000001
PSLSS_PVMOD     = 00000002
PSLSV_PVMOD     = 00000016
PTR_B_LEVEL     = 00000004
PTR_B_NUMBER    = 00000005
PTR_B_PARMCNT   = 00000006
PTR_B_VALUE     = 00000000
PTR_C_LENGTH    = 0000000C
PTR_K_COLON     = 00000002
PTR_K_CMDQUAL   = 00000000
PTR_K_LENGTH    = 0000000C
PTR_K_PARAMETR  = 00000003
PTR_L_DESCR     = 00000000
PTR_L_ENTITY    = 00000008
PTR_V_FLAGS     = 00000014
PTR_V_NEGATE    = 00000014
READ_AST       = 000013BA
RESTORE_CONTEXT = 000006F1
RETURNED_MESSAGE = 00000E36
SPAWN_EXIT      = 000006C2
SPAWN_PROCESS   = 00000544
SPWN_B_ACMODE   = 0000000E
SPWN_B_CONTINUE = 000000A5
SPWN_B_EFN      = 0000000F
SPWN_B_PROMPTLEN = 000000A2

```

R 02
R 02
R 02
R 02

SPAWN
Symbol table

- MULTI-PROCESSING COMMANDS

L 16

16-SEP-1984 00:17:05 VAX/VMS Macro V04-00
4-SEP-1984 23:43:20 [DCL.SRC]SPAWN.MAR;1

Page 66
(27)

```

SPWN_C_LENGTH      00000006
SPWN_G_PROMPT      000000A6
SPWN_G_QUOTAS      00000060
SPWN_K_LENGTH      00000006
SPWN_L_ASTADR      0000004C
SPWN_L_ASTPRM      00000050
SPWN_L_IMAGCNT      0000005C
SPWN_L_LINK        00000000
SPWN_L_OUTOFBAND    00000058
SPWN_L_PRI8        00000048
SPWN_L_STATUS      00000044
SPWN_L_STSADR      00000054
SPWN_L_SUBPID      00000040
SPWN_M_CLI          = 00002000
SPWN_M_CLISYM      = 00000020
SPWN_M_KEYPAD      = 00001000
SPWN_M_LOG          = 00000001
SPWN_M_LOGNAM      = 00000040
SPWN_M_TABLE        = 00008000
SPWN_M_WAIT         = 00000004
SPWN_Q_CLI          000000C6
SPWN_Q_CMDSTR       00000030
SPWN_Q_INPUT        00000020
SPWN_Q_IOSB         00000038
SPWN_Q_MBXNAM       00000010
SPWN_Q_OUTPUT       00000028
SPWN_Q_PRCNAM       00000018
SPWN_Q_TABLE        000000CE
SPWN_T_PROCESS      00000092
SPWN_V_ACTIVE       = 00000007
SPWN_V_AUTOLOGO     = 00000003
SPWN_V_CLI          = 0000000D
SPWN_V_CLISYM       = 00000005
SPWN_V_INPUT        = 0000000A
SPWN_V_KEYPAD       = 0000000C
SPWN_V_LOG          = 00000000
SPWN_V_LOGNAM       = 00000006
SPWN_V_MODE         = 00000004
SPWN_V_NOCTX        = 0000000E
SPWN_V_NOTIFY       = 00000008
SPWN_V_OUTPUT       = 0000000B
SPWN_V_PRCNAM       = 00000001
SPWN_V_PROMPT       = 00000009
SPWN_V_TABLE        = 0000000F
SPWN_V_WAIT         = 00000002
SPWN_W_CHAN         0000000A
SPWN_W_FLAGS        0000000C
SPWN_W_PMPTCTRL     000000A3
SPWN_W_SIZE         00000004
SPWN_W_UNIT         00000008
SSS_DUPLNAM        = 00000094
SSS_ENDOFFILE       = 00000870
SYS_MAXBUF         = 0000104F
SYM_B_FLAGS         0000000B
SYM_B_NONUNIQUE     0000000B
SYM_B_TYPE          0000000A
SYM_K_BINARY        = 00000002

```

```

SYM_K_KEYPAD       = 00000004
SYM_K_PERM         = 00000001
SYM_K_STRING       = 00000000
SYM_L_BL           00000004
SYM_L_FL           00000000
SYM_T_SYMBOL       0000000C
SYM_W_SIZE         00000008
SYSSASSIGN         ***** GX 02
SYSSBRKTHRU        ***** GX 02
SYSSCANCEL         ***** GX 02
SYSSCLREF          ***** GX 02
SYSSCREMBX         ***** GX 02
SYSSCREPRC         ***** GX 02
SYSSDASSGN         ***** GX 02
SYSSDCLAST         ***** GX 02
SYSSDELPRC         ***** GX 02
SYSSFAO            ***** X 02
SYSSGETCHN         ***** GX 02
SYSSGETDVIW        ***** GX 02
SYSSGETJPIW        ***** GX 02
SYSSGETSYIW        ***** GX 02
SYSSHIBER          ***** GX 02
SYSSINPUT          0000003A R 02
SYSSOUTPUT         00000044 R 02
SYSSPARSE          ***** GX 02
SYSSQIOW           ***** GX 02
SYSSSEARCH         ***** GX 02
SYSSSETEF          ***** GX 02
SYSSSYSTEM         0000004F R 02
SYSSWAKE           ***** GX 02
TERMINATION_AST    00001461 R 02
TMBX_B_REFS        00000008
TMBX_C_LENGTH      00000010
TMBX_C_MAXREFS     = 00000004
TMBX_K_LENGTH      00000010
TMBX_L_LINK        00000000
TMBX_L_PRC         0000000C
TMBX_W_CHANNEL     00000006
TMBX_W_SIZE        0000000A
TMBX_W_UNIT        00000004
TT2SV_DCL_MAILBX   = 00000009
VERIFY_INPUT       00000710 R 02
VERIFY_OUTPUT      000008EF R 02
WRITE_AST          00001329 R 02
WRITE_CONTEXT      00000E80 R 02
WRITE_MAILBOX      0000111B R 02
WRITE_SYMBOLS      00001149 R 02
WRK_B_CMDOPT       FFFFFFFC3
WRK_B_MAXPARM      FFFFFFFD0
WRK_B_MINPARM      FFFFFFFD1
WRK_B_PARMCNT      FFFFFFFCE
WRK_B_PARMSUM      FFFFFFFCF
WRK_B_RECALLCNT    FFFFFFFC5
WRK_B_VALLEV       FFFFFFFC4
WRK_B_VERBTYP      FFFFFFFC2
WRK_C_INPBUFFSIZ   = 00000100
WRK_C_LENGTH       FFFFF486

```

SPAWN
Symbol table

- MULTI-PROCESSING COMMANDS

M 16

16-SEP-1984 00:17:05 VAX/VMS Macro V04-00
4-SEP-1984 23:43:20 [DCL.SRC]SPAWN.MAR;1

Page 67
(27)

```

WRK_G_BUFFER      FFFFFFF492
WRK_G_INPBUF      FFFFFFF896
WRK_G_RESULT      FFFFFFF9B6
WRK_K_LENGTH      FFFFFFF486
WRK_L_CHARPTR     FFFFFFF48E
WRK_L_DISALLOW    FFFFFFFFE6
WRK_L_ERRORRTN    FFFFFFF9AE
WRK_L_EXPANDPTR   FFFFFFF486
WRK_L_IMAGE       FFFFFFFFE2
WRK_L_MARKPTR     FFFFFFF48A
WRK_L_PAROUT      FFFFFFFFD2
WRK_L_PMPTADDR    FFFFFFF9A2
WRK_L_PROMPTRTN   FFFFFFF9A6
WRK_L_PROPTR      FFFFFFFFC6
WRK_L_QUABLK      FFFFFFFCA
WRK_L_READRTN     FFFFFFF9AA
WRK_L_RECALLPTR   FFFFFFFFEA
WRK_L_RSLND       FFFFFFFB6
WRK_L_RSLNXT      FFFFFFFBA
WRK_L_SAVAP       FFFFFFFF8
WRK_L_SAVFP       FFFFFFFFC
WRK_L_SAVSP       FFFFFFFF4
WRK_L_SIGNALRTN   FFFFFFFD6
WRK_L_SPECRTN     FFFFFFF9B2
WRK_L_TAB_VEC     FFFFFFFDE
WRK_L_VERB        FFFFFFFBE
WRK_W_FLAGS       FFFFFFFF0
WRK_W_FLAGS2      FFFFFFFF2
WRK_W_IMGCHAN     FFFFFFFEE
WRK_W_PMPTLEN     FFFFFFF9E
_SS_              = 000000EF

```

! Psect synopsis !

PSECT name	Allocation	PSECT No.	Attributes														
. ABS .	00000000 (0.)	00 (0.)	NOPIC	USR	CON	ABS	LCL	NOSHR	NOEXE	NORD	NOWRT	NOVEC	BYTE				
\$ABSS	FFFFFFFFC (0.)	01 (1.)	NOPIC	USR	CON	ABS	LCL	NOSHR	EXE	RD	WRT	NOVEC	BYTE				
DCL\$ZCODE	00001619 (5657.)	02 (2.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	NOWRT	NOVEC	BYTE				

! Performance indicators !

Phase	Page faults	CPU Time	Elapsed Time
Initialization	16	00:00:00.05	00:00:01.03
Command processing	97	00:00:00.61	00:00:07.31
Pass 1	835	00:00:43.45	00:02:15.40
Symbol table sort	0	00:00:05.26	00:00:14.71
Pass 2	399	00:00:09.91	00:00:32.77
Symbol table output	0	00:00:00.40	00:00:00.95
Psect synopsis output	0	00:00:00.02	00:00:00.38
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	1347	00:00:59.71	00:03:12.55

The working set limit was 2700 pages.
240218 bytes (470 pages) of virtual memory were used to buffer the intermediate code.
There were 180 pages of symbol table space allocated to hold 3272 non-local and 181 local symbols.
2766 source lines were read in Pass 1, producing 32 object records in Pass 2.
96 pages of virtual memory were used to define 72 macros.

! Macro library statistics !

Macro library name	Macros defined
-----	-----
\$255\$DUA28:[SYSLIB]SYSBLDMLB.MLB;1	0
\$255\$DUA28:[DCL.OBJ]DCL.MLB;1	14
\$255\$DUA28:[SYS.OBJ]LIB.MLB;1	3
\$255\$DUA28:[SYSLIB]STARLET.MLB;2	46
TOTALS (all libraries)	63

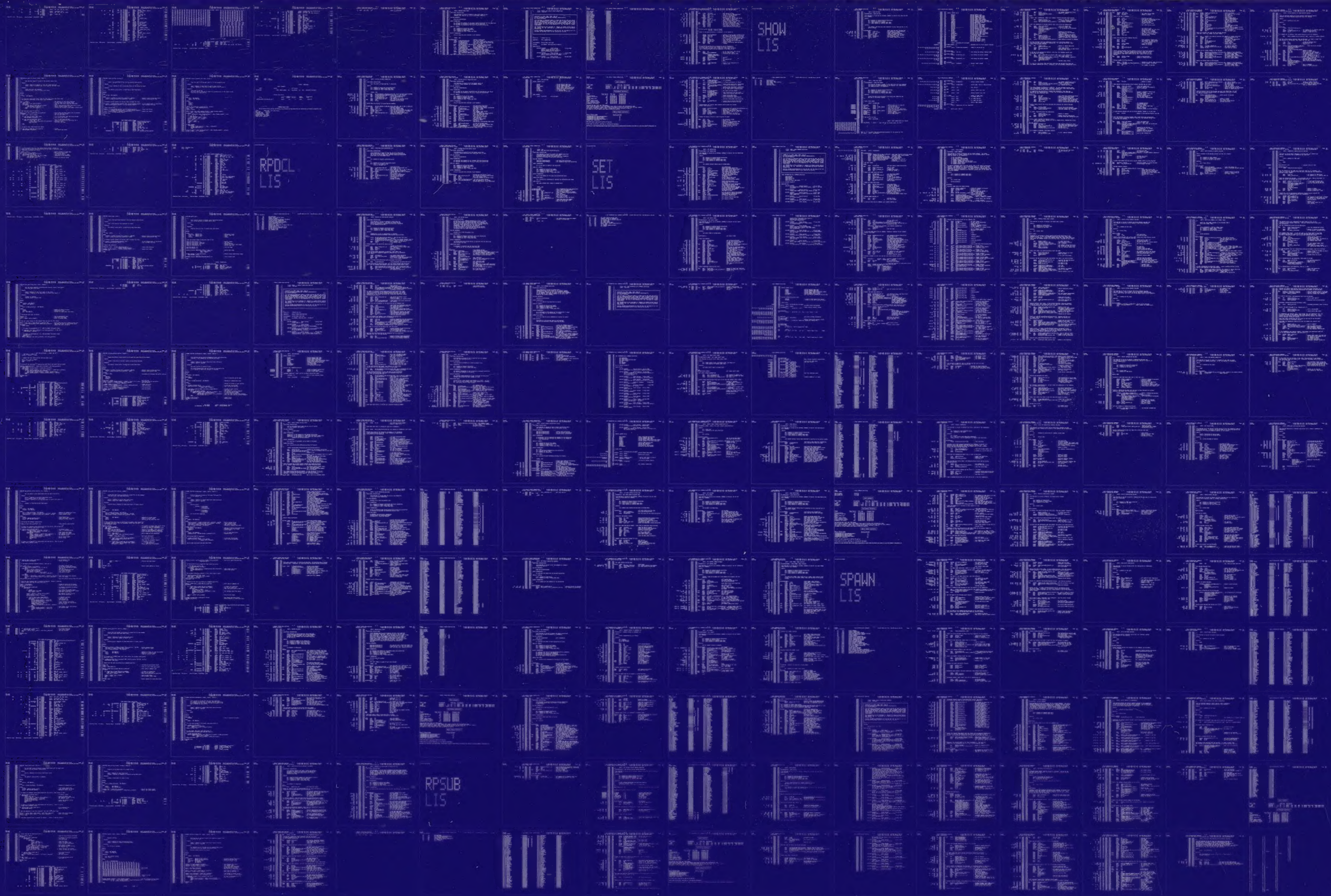
3762 GETS were required to define 63 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS\$:SPAWN/OBJ=OBJ\$:SPAWN MSRC\$:SPAWN/UPDATE=(ENH\$:SPAWN)+EXECML\$/LIB+LIB\$:DCL/LIB+SYSS\$LIBRARY:SYSBLDMLB/LIB

0073 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY



0074

AH-BT13A-SE
 VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY